目　　录

HY—8130 八路高速数据采集板 ····· 4

一、概述 ····· 4
二、性能及技术指标 ····· 5
2.1 性能 ····· 5
2.2 应用 ····· 5
2.3 技术指标 ····· 5
2.4 物理特征及环境工作条件 ····· 6
三、工作原理 ····· 6
3.1 原理框图 ····· 6
3.2 工作原理概述 ····· 6
四、主要元件位置图、信号输入 / 输出插座和跳线选择定义 ····· 7
4.1 主要元件位置图 ····· 7
4.1.1 HY—8130 板主要元件位置图 ····· 7
4.1.2 HY—8130板的背板主要元件位置图 ····· 8
4.2 信号输入 / 输出插座定义 ····· 8
4.3 跳线选择 ····· 8
4.3.1 板基地址跳线选择 ····· 8
4.3.2 触发信号选择跳线 XF3 ····· 9
4.3.3 中断申请信号选择跳线 XF13 ····· 9
4.3.4 模拟信号输入范围选择（背板）跳线 ····· 9
4.3.5 DA输出范围选择跳线XF9 ····· 10
五、寄存器描述及板内地址分配 ····· 10
5.1 板内地址分配 ····· 10
5.2 程序切换板上存储器地址（板基地址+0读） ····· 10
5.3 模拟信号输入通道控制寄存器（板基地址+0写） ····· 10
5.4 A/D 转换结果寄存器（板基地址+2读） ····· 10
5.5 D/A 输出寄存器（板基地址+2写） ····· 10
5.6 数据采集结束标志寄存器（板基地址+4读） ····· 12
5.7 板控制字寄存器（板基地址+4写） ····· 12
5.8 工作就绪状态寄存器（板基地址+6读） ····· 13
5.9 8254定时/计数器0数据寄存器（板基地址+8写） ····· 13
5.10 8254定时/计数器1数据寄存器（板基地址+10写） ····· 13
5.11 8254定时/计数器2数据寄存器（板基地址+12写） ····· 13
5.12 8254控制字寄存器（板基地址+14写） ····· 13
六、安装、拆除方法 ····· 13

6.1 安装步骤 ····· *13*
6.2 拆除步骤 ····· *14*
七、软件编程 ····· 14
7.1 可编程定时/计数器8254编程描述 ····· *14*
7.2 8254 在HY—8130中的应用 ····· *16*
7.3 AD板触发方式 ····· *18*
7.3.1 软件触发 ····· *18*
7.3.2 提前触发和滞后触发 ····· *19*
7.3.3 外触发工作方式 ····· *22*
7.4 外部时钟启动AD采样工作方式 ····· *23*
7.5 多块HY—8130板同步工作方式 ····· *24*
7.6 编程实例 ····· *24*
7.6.1 软件触发工作方式编程实例 ····· *24*
7.6.2 提前触发工作方式编程实例 ····· *25*
7.6.3 滞后触发工作方式编程实例 ····· *26*
7.6.4 中断方式采样编程实例 ····· *27*
7.6.5 外部时钟启动A/D采样编程实例 ····· *28*
7.6.6 外部触发编程实例 ····· *29*
八、应用注意事项 ····· 30
8.1 注意事项 ····· *30*
九.校准 ····· 30
9.1 校准准备 ····· *30*
9.2 A/D校准 ····· *31*
十、I/O 地址分配 ····· 32
附录1：HY—8130测试软件使用说明 ····· 33
一、运行所需环境 ····· 33
1.1 硬件环境 ····· *33*
1.2 软件环境 ····· *33*
二、软件的组成和安装 ····· 33
三、界面简介 ····· 33
3.1 初始画面介绍 ····· *33*
3.2 系统菜单介绍 ····· *34*
3.3 各功能按钮介绍 ····· *35*
3.4 各面板功能介绍 ····· *35*
3.4.1 示波面板 ····· *35*
3.4.2 万用表面板 ····· *36*
四、操作方式 ····· 36
4.1 使用鼠标 ····· *36*

4.2 使用键盘 …… 37
附录2：HY—8130驱动函数使用说明 …… 38
一、驱动函数库 …… 38
二、函数说明 …… 39
三、驱动函数的使用方法 …… 43
3.1 用C语言编程调用 …… 43
3.2 用BASIC语言编程调用 …… 45

HY—8130 八路高速数据采集板

一、概述

HY—8130 板是IBM—PC / AT 总线兼容的12位高速数据采集板，采样通过率可达到1MHz，并具有多种触发方式。HY—8130 板采用四层印刷线路板和背板式结构，可直接插入IBM—PC　AT总线兼容的计算机内的任意总线扩展槽中，构成各种瞬态信号采集、振动信号测量及频谱分析等领域的高速数据采集系统。

HY—8130板初始化后，可自动完成模拟量电压信号采集过程，无需主机干预，并将采集结果存入板上存储器中。板上存储器容量可达512K字，能满足高速、大数据量采集应用的要求。尤其适合在Windows环境下工作的数据采集系统中使用。

HY—8130板上安装有12Bit分辨率的A/D转换器，为用户提供了
八路单端模拟输入通道，A/D转换器可以选择模拟输入信号范围是：0～4.096V、±2.048V、±5.016V。

HY—8130板支持软件触发、提前触发、滞后触发和外触发四种触发工作方式，支持定时启动和外部时钟启动两种A/D电路启动转换工作方式，支持软件查询和中断申请数据传送工作方式。

HY—8130板支持多块8130板同步采集工作方式。

HY—8130板可外接 HY—822 四路同步采保输入滤波端子板。并有16 路锁存式数字量D/O输出。

HY—8130 板以其存储器配置的不同分为A、B两种型号。用户可根据需要选用。

HY—8130A 型存储器容量为：128K 字

HY—8130B 型存储器容量为：512K 字

二、性能及技术指标

2.1 性能

- 8 路单端模拟信号输入通道
- A/D转换分辨率：12 Bit
- 模拟量电压输入范围：

  0～4.096V、±2.048V、±5.016V
- 支持软件触发、提前触发、滞后触发、外触发四种A/D触发方式
- 支持定时启动、外部时钟启动等A/D转换电路启动方式
- 支持软件查询、中断申请等数据传输方式
- 板上存储器容量：128K 字（A型）；512K 字（B型）
- 16 路锁存式数字量输出通道
- 板上时钟频率：8 MHZ
- 程控阈值电平输出范围：±5V

2.2 应用

- 瞬态信号采集
- 振动信号测量
- 噪声分析
- 数字存储示波器
- 频谱分析

2.3 技术指标

模拟信号输入部分

- 模拟输入通道数：8 路单端模拟信号输入
- 模拟输入电压范围：0～4.096V、±2.048V、±5.016V
- 模拟输入阻抗：＞100 KΩ
- 放大器建立时间：350 ns（精度0.01％）

A/D转换电路部分

- A/D分辨率：12 Bit（4096）
- 非线性：±1LSB（最大）
- 转换时间（含采样保持时间）：1μS
- A/D板转换精度：±0.1%
- 程控阈值电平触发精度：±10mV
- 程控阈值电平输出建立时间：10μS
- 程控阈值电平输出范围：±5V

数字量部分

- 板上存储器容量：128K 字（A型）；512K 字（B型）
- 数字量输出通道数：16 Bit（并行锁存、TTL 电平）
- 输出电平特性：TTL 电平兼容

  输出高电平：$V_{OH}\geqslant 2.8V$

  输出高电平电流：$I_{OH}\leqslant 400\mu A$

  输出低电平：$V_{OL}\leqslant 0.5V$

  输出低电平电流：$I_{OL}\leqslant 2mA$

中断申请部分

• 中断申请通道数：1 路
• 中断申请级别：IRQ10、IRQ11、IRQ12、IRQ14、IRQ15

2.4 物理特征及环境工作条件

• 外形尺寸：21cm×12cm×2.5cm
• 工作温度范围：10℃～＋50℃
• 贮藏温度范围：－25℃～＋85℃
• 湿度范围：90％（不结露）
• 主机电源功耗：＋5V，1A

注：以上技术指标未注明者均为典型值。

三、工作原理

3.1 原理框图

图3.1为HY—8130 板的工作原理框图。

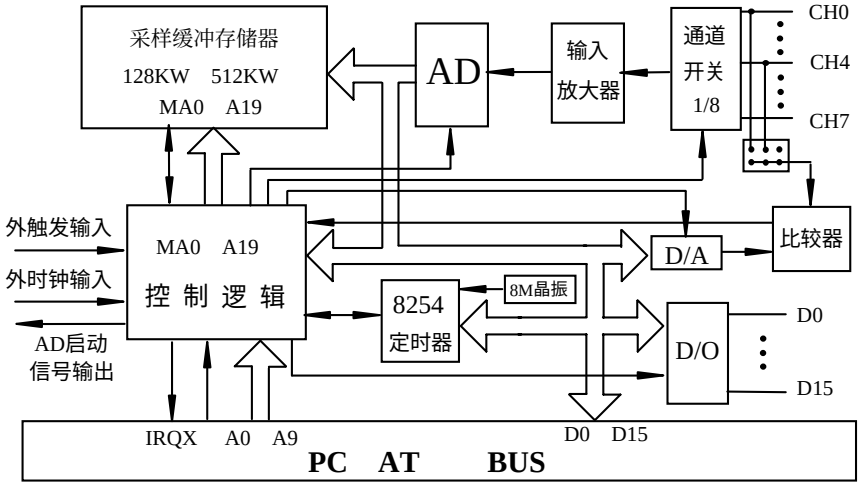


图3.1 HY—8130 板工作原理框图

3.2 工作原理概述

HY—8130板主要由模 / 数转换（A/D）电路、存储器电路、触发电路、控制逻辑电路、数字量输出（DO）电路及PC / AT总线接口电路等六部分组成。

HY—8130板对模入信号的采集和捕捉，是通过使用不同的触发工作方式实现的。A/D转换电路的触发方式分为软件触发、提前触发、滞后触发和外触发四种。

软件触发工作方式，由程序指令启动采集过程。

提前触发或滞后触发工作方式，由阈值电平触发电路（D/A和比较器组成）将模拟输入通道CH0或通道CH4的被测信号电平，与板上由D/A设定的阈值电平进行电平比较，当模入电平达到阈值电平时，产生触发信号，启动采样过程。通过对采样数据长度计数器的编程，实现对触发信号来到前后一段时间内各模入通道的信号采集和捕捉。

外触发工作方式，是由外部触发脉冲信号启动采集过程。每来一个外触发脉冲信号，A/D对设定的模入通道范围内各模入信号采集一遍。采集的数据量达到采样数据长度计数器设定的长度后，自动停止A/D转换。

在采样工作开始前，用户通过软件编程，对AD板控制寄存器进行初始化。其内容包括设置采样频率、采样数据长度、触发方式、触发阈值电平等参数。触发信号来到，A/D板按初始化设置开始对模入信号进行采集。外部的八路被测模入信号，经过通道开关选择，自动在选定的通道范围内，把不同通道的信号分时送入A/D转换

器。转换后的数据，在采样长度计数器控制下，把数据存入板上的采样缓冲存储器中暂存，再由程序查询采样结束标志寄存器或向主机申请中断，从采样缓冲存储器向主机传送采集结果。

A/D电路启动转换工作方式分为定时启动和外部时钟启动两种。定时启动AD工作方式使用板上8254定时/计数器 0 的输出作为A/D启动信号。外时钟工作方式是直接将外部时钟信号作为A/D启动信号。一但程序命令允许外部时钟信号开始工作，控制逻辑就按照初始化设定好的方式，对被测模入电压信号进行A/D转换，并将转换结果放入板上存储器中。

HY—8130板支持多块8130板同步采集工作方式。可以多块HY—8130板插入同一台PC机中，从中任选一块作为主板，其他作为从板。由主板将其AD启动信号输出，给从板的外部时钟输入端，引入AD启动信号，构成同步多路高速数据采集系统。

使用同步多采保端子板时，在每个模入通道采样循环中，自动为同步多采保信号插入一段建立时间，其长度等于一个AD转换周期。

板上使用8254的定时/计数器0 对8M 晶振进行程控分频，其输出用来控制AD转换周期，使用定时/计数器1和2 级联后，构成采样长度计数器。

数字量输出电路，使用一条I/O写指令，既可将16位开关量数据锁存输出。

关于工作方式的使用方法，希望用户按有关章节所提供的工作顺序、计算公式和应用举例进行编程。

四、主要元件位置图、信号输入／输出插座和跳线选择定义

4.1 主要元件位置图

4.1.1 HY—8130 板主要元件位置图

图4.1.1为HY—8130板的主要元件位置图，此元件位置图上的开关和跳线设置为出厂标准设置。板基地址：280H；中断级别：IRQ10；触发信号通道：CH0；触发阈值电平范围：±5V；

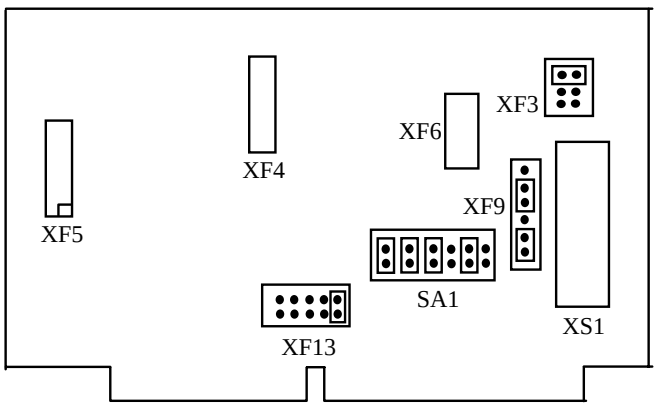


图4.1.1　HY—8130 板主要元件位置图

XF3：触发信号通道选择跳线

XF4、XF6：从HY—8130主板到背板的信号传输插座

XF5：16位数字量输出插座

XF9：阈值电平范围选择插座

SA1：板基地址选择跳线

XS1：模入信号、触发信号输入，AD启动信号输出插座

XF13：中断申请信号选择跳线

4.1.2 HY—8130板的背板主要元件位置图

RP1：AD满度调整
RP2：AD零点调整
XF1～XF4：模入量程范围选择（出厂状态±5.016V）
XS1、XS2：背板到主板信号引线插座

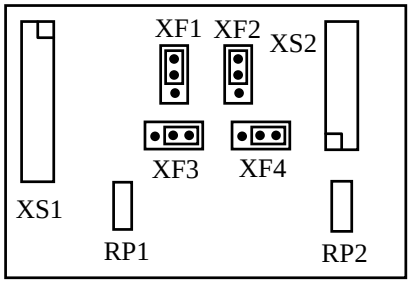
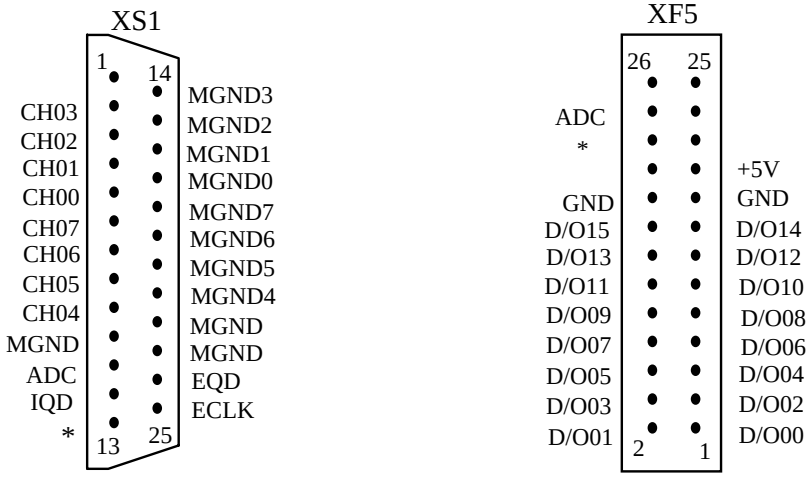


图4.1.2 背板主要元件位置图

注：图上未列出的插座是留给生产、调试设置的。与用户使用无关！请用户不要改变出厂状态！

4.2 信号输入／输出插座定义

图4.2为 模拟信号、触发信号输入／数字量信号输出插座定义图。



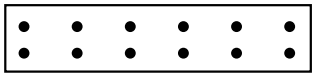
4.2 模拟信号、触发信号输入／数字量信号输出插座定义

- CH00～CH07：模拟信号输入正端
- MGND0～MGND7：模拟信号输入地端
- IQD：内部信号输出端
- EQD：外部触发信号输入端
- ECLK：外部TTL时钟脉冲信号输入端
- ADC：AD启动信号输出
- D/O00～D/O15：数字量信号输出端
- GND：数字地
- MGND：模拟地端
- ＋5V：电源
- ＊：保留

4.3 跳线选择

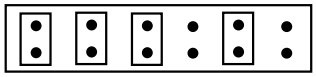
4.3.1 板基地址跳线选择

HY—8130的板基地址通过对SA1的跳线选择来设置，板基地址可设置成200H~3F8H范围内地址A4～A9的任意二进制码组合，
HY—8130将占用基地址起的连续16 个I/O地址。板基地址选择跳线SA1示意及举例说明如图4.3.1：



地址线：	A4	A5	A6	A7	A8	A9
十进制：	16	32	64	128	256	512
十六进制：	10	20	40	80	100	200

例



A4	A5	A6	A7	A8	A9
板基地址	512	128	640		
或	0200H	80H	280H		

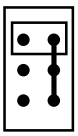
图4.3.1 板基地址选择跳线定义及举例说明

板基地址范围512～1016（0200H~03F8H），插上跳线为无效，拔下跳线为有效。板基地址等于所有有效位数之和。

4.3.2 触发信号选择跳线 XF3

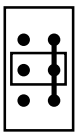
在使用提前触发或滞后触发工作方式时，是将模拟量输入通道0（CH0）或通道4（CH4）的被测 信号电平，与板上的D/A设定的阈值电平进行比较，当模入电平达到阈值电平时，启动采集过程。通过对采样数据长度计数器的编程，实现对触发信号来到前后一段时间内各模入通道的信号捕捉和采集。可以通过对XF3的跳线选择模入通道。

XF3



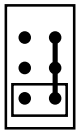
选择模入通道0

XF3



选择模入通道4

XF3



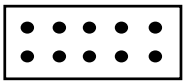
未使用

4.3.3 中断申请信号选择跳线 XF13

跳线器XF13用来选择中断级别，可选择主机中断级别为：IRQ10～IRQ12、IRQ14、IRQ15。选择方式见下图：

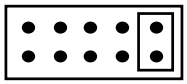
例如：选择IRQ10

XF13



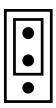
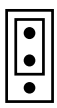
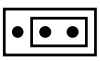
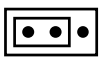
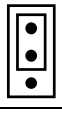
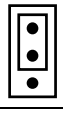
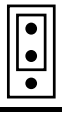
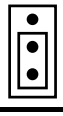
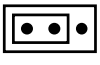
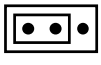
IRQ14 IRQ15 IRQ12 IRQ11 IRQ10

XF13

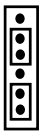


IRQ14 IRQ15 IRQ12 IRQ11 IRQ10

4.3.4 模拟信号输入范围选择（背板）跳线

量程	XF1	XF2	XF3	XF4
±2.048V				
±5.016V				
0～4.096V				

4.3.5 DA输出范围选择跳线XF9

XF9  DA 输出范围为 5V

五、寄存器描述及板内地址分配

5.1 板内地址分配

表5.1为HY—8130板内地址分配表，详细使用参见第七章软件编程。

表5.1 HY—8130板内地址分配表

地址	写	读
板基地址＋0	模入通道控制寄存器	程序切换板上存储器地址
板基地址＋2	D/A 转换数据寄存器	A/D 转换数据寄存器
板基地址＋4	板控制字寄存器	采集结束标志寄存器
板基地址＋6	16 位D/O输出寄存器	工作就绪状态寄存器
板基地址＋8	8254数据寄存器	无
板基地址＋10	8254数据寄存器	无
板基地址＋12	8254数据寄存器	无
板基地址＋14	8254 命令寄存器	无

5.2 程序切换板上存储器地址（板基地址＋0读）

程序对板基地址＋0 口进行一次读操作，HY—8130板上存储器地址就自动＋1。

5.3 模拟信号输入通道控制寄存器（板基地址＋0写）

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
×	×	×	×	×	×	BL	BL	CHH3	CHH2	CHH1	CHH0	CHL3	CHL2	CHL1	CHL0

其中： ×：未使用

CHL0～CHL3：A/D循环采样的首通道

CHH0～CHH3：A/D循环采样的末通道

BL：保留

注意： 只要CHL和CHH设定后，每进行一次AD转换，模入通道在 CHL和CHH之间自动循环加1。

5.4 A/D 转换结果寄存器（板基地址＋2读）

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
BZ	×	×	×	AD11	AD10	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0

其中： ×：未使用

AD0～AD11：为AD 转换结果的12 位数据。

BZ：通道循环结束标志位。

BZ＝0：数据为循环的末通道（CHH）采样数据。

BZ＝1：数据为其它通道采样数据

5.5 D/A 输出寄存器（板基地址＋2写）

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

×	×	×	×	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

其中： ×：未使用

DB0～DB11：写入触发阈值电平的12位DA模拟输出数据，写 入DA数据的同时，启动DA将模拟数据输出。

HY—8130板的D/A转换器的数据格式如下表所示：

二进制数	十六进制数	模拟输出
0000 0000 0000	000	V_{INL}
0000 0000 0001	001	$\frac{V_{INH}-V_{INL}}{4096}+V_{INL}$
1000 0000 0000	800	$\frac{(V_{INH}-V_{INL})\times 2048}{4096}+V_{INL}$
1111 1111 1111	FFF	$\frac{(V_{INH}-V_{INL})\times 4095}{4096}+V_{INL}$

其中： V_{INL} 为模拟输出负满度（或零点）值。

V_{INH} 为模拟输出正满度值。

5.6 数据采集结束标志寄存器（板基地址＋4读）

D7	D6	D5	D4	D3	D2	D1	D0
×	×	×	×	×	×	×	TB

其中： ×：未使用

TB＝1：完成采样长度计数器设定的AD转换次数，已经把数据放在板上采样缓冲存储器中

TB＝0：未完成设定的AD转换次数

5.7 板控制字寄存器（板基地址＋4写）

在板状态控制字中，D0～D7是状态寄存器控制位，对板基地址＋4写入后，不重新设定，状态将不会改变。D8、D9两位仅在控制字写操作期间有效。

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
×	×	×	×	×	×	K9	K8	K7	K6	K5	K4	K3	K2	K1	K0

其中： ×：未使用

K0：采样长度计数器控制位。

K0＝1：允许计数。

K0＝0：禁止计数。

K1：外触发允许控制位。

K1＝1：允许外触发电路工作。

K1＝0：禁止外触发电路工作。

K2：中断允许控制位。

K2＝1：允许发出中断申请信号。

K2＝0：禁止发出中断申请信号。

K3：板上存储器工作方式控制位。

K3＝1：允许AD转换结果写入板上存储器。

K3＝0：允许从板上存储器中读AD转换结果。

K4：阈值电平触发允许控制位。

K4＝1：允许阈值电平触发电路工作。

K4＝0：禁止阈值电平触发电路工作。

K5：阈值电平触发工作方式控制位。

K5＝1：外部模入信号上升沿交越阈值电平，A/D触发电　路开始工作。

K5＝0：外部模入信号下降沿交越阈值电平，A/D触发电　路开始工作。

K6：外部脉冲启动AD转换允许控制位。

K6＝1：允许外部时钟脉冲启动AD工作。

K6＝0：禁止外部时钟脉冲启动AD工作。

K7：外部同步多采保控制信号输出允许控制位。

K7＝1：允许同步多采保控制信号生成。自动在每次模入　通道采样循环开始之前增加一个AD转换周期，为外部采样/保持电路提供信号建立时间。

K7＝0：禁止同步多采保控制信号生成。

K8：板上存储器地址复位控制。

K8＝1：存储器地址复位；

K8＝0：未使用。

K9：采集结束标志复位控制位。

K9＝1：清AD采集结束标志寄存器。仅在外触发或外时钟工作方式下使用。其它工作方式自动清除AD采集结束标志寄存器。

K9＝0：未使用。

5.8 工作就绪状态寄存器（板基地址＋6读）

对HY—8130 初始化完成，准备好进行AD转换后，对板基地址＋6口进行一次读操作，使HY—8130板按初始化设置的内容开始工作。

5.9 8254定时/计数器0数据寄存器（板基地址＋8写）

8254定时/计数器0 ，在HY—8130板上作为A/D采样频率数据寄存器使用。16位A/D采样频率数据，分两次写入。具体使用方法请参见7.2 节。

5.10 8254定时/计数器1数据寄存器（板基地址＋10写）

8254定时/计数器1，在HY—8130板上作为A/D采样数据长度计数器的低16位数据寄存器。16位数据，分两次写入。具体使用方法请参见 7.2 节。

5.11 8254定时/计数器2数据寄存器（板基地址＋12写）

8254定时/计数器2，在HY—8130板上作为A/D采样长度数据计数器的高16位数据寄存器。16位数据，分两次写入。具体使用方法请参见 7.2 节。

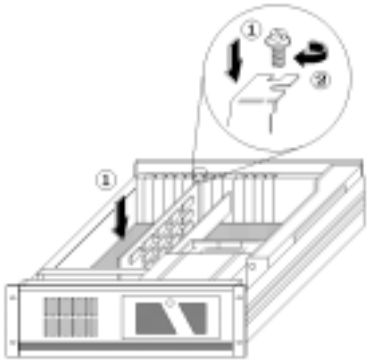
8254 的定时/计数器1和2 已在硬件上级连起来，构成一个32位数据寄存器，作为A/D采样长度数据计数器。

5.12 8254控制字寄存器（板基地址＋14写）

8254定时/计数器 的控制字寄存器，是8位寄存器，用字节操作指令写入。控制字寄存器使用方法请参见7.1 节。

六、安装、拆除方法

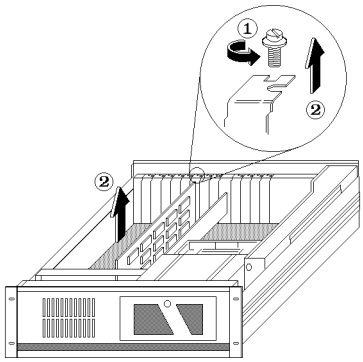
HY—8130板禁止带电插拔。接插和拔下引线插头，开关的设置及跳线器选择都必须在断电的情况下进行。



6.1 安装步骤

①把模板垂直对准任一扩展槽，以手向下用力将该板插入扩展槽。

②拧紧螺钉，连接输入/输出电缆。



6.2 拆除步骤

①断开输入/输出信号电缆，拧下螺钉。
②双手向上垂直用力，将该板从扩展槽中拔出。

七、软件编程

7.1 可编程定时/计数器8254编程描述

有关8254详细情况请参见8254技术手册或有关资料。
在使用8254内部计数器前，必须先向8254内部控制字寄存器写入控制字和写入计数器预置值。
控制字寄存器格式如下：

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

各位定义如下：
BCD：计数器计数方式选择，可采用二进制或BCD码。
M2、M1、M0：计数器工作方式选择，可有六种工作方式，具体 含义见下表。

M2	M1	M0	方式
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5

BCD	计数类型
0	二进制计数
1	BCD码计数

RL1、RL0计数器读写操作长度选择，以决定对计数器进行装入或读出是双字节还是单字节。SC1、SC0选择计数器0、1或2。

RL1	RL0	操作类型
0	0	计数器锁存操作
0	1	只读/写低字节
1	0	只读/写高字节
1	1	先读/写低字节 后读/写高字节

SC1	SC0	选择计数器
0	0	计数器0
0	1	计数器1
1	0	计数器2
1	1	非法

当对8254写入控制字后，就要给计数器赋初值了。当控制字
D0＝0时，即二进制计数，初值可在0000H~FFFFH之间选择，当控制字D0＝1时，则装入计数器的初值应选十进制方式，其值可在
0000～9999十进制数之间选择，但无论何种计数方式，当初值为
0000时，计数器的计数值最大。

例如：在HY—8130板上，下面几条指令将8254的三个定时／计数器设置成二进制计数、计数器双字节读写操作。设置工作方式分别为：工作方式0、工作方式2。

```
int   baseA=640;                /*板基地址＝640*/
outp( baseA+14,0xb0);           /*设置定时／计数器2为工作方式0 */
outp( baseA+14,0x74);           /*设置定时／计数器1为工作方式2 */
outp( baseA+14,0x34)            /*设置定时／计数器0为工作方式2 */
```

工作方式说明

方式0—计数结束产生中断方式

当写入方式0控制字后，计数器输出立即变成低电平，当赋初值后，计数器马上开始计数，并且输出一直保持低电平，当计数结束时变成高电平，并且一直保持到重新装入初值或复位时为止。当控制字中D5D4＝11时，在写入低字节后计数器还不计数，当写入高字节后，计数器才开始计数，如果对正在做计数的计数器装入一个新值，则计数器又从新装入的计数值开始，重新作减量计数。可用门控端GATE控制计数，当GATE＝0时，禁止计数，当GATE＝1时，允许计数。

方式1—可编程单次脉冲方式

该方式要在门控信号GATE作用下工作。当装入计数初值N之后，要等GATE由低变高，并保持高时开始计数，此时输出OUT变成低电平，当计数结束时，输出变成高电平，即输出单次脉冲的宽度由装入的计数初值N来决定。当计数器减量计数未到零时，又装入一个新的计数值N1，则这个新值，不会影响当前的操作，只有原计数值减到零且有一个GATE上升沿时，计数器才从N1开始计数。如当前操作还未完，又有一次GATE上升沿时，则停止当前计数，又重新从N1开始计数，这时输出单次脉冲就被加宽。

方式2—频率发生器方式

在该方式下，计数器装入初始值，开始工作后，输出端将不断输出负脉冲，其宽度等于一个时钟周期，两负脉冲间的时钟个数等于计数器装入的初始值。在方式2中门控信号相当于复位信号，当GATE＝0时，立即强迫输出为高电平，当GATE＝1时，便启动一次新的计数周期，这样可以用一个外部控制逻辑来控制GATE，从而达到同步计数的作用。当然计数器也可以用软件控制GATE而达到同步控制目的。

方式3—方波频率发生器方式

与方式2类似，当装入一个计数器初值N后，在GATE信号上升沿启动计数，定时/计数器此时作减2计数，在完成前一半计数时，输出一直保持高电平，而在进行后一半计数时，输出又变成低电平。若装入的数N为奇数，则在（N＋1）/2个计数期间，输出保持高电平。在
（N－1）/2个计数期间，输出保持低电平。若在一次计数期间，将一个新的初值装入计数器，那么在当前的计数发生跳变时，计数器马上又按新的计数开始计数。

方式4—软件触发选通方式

用控制字设置该方式后，输出即变为高电平，在GATE＝1时，计数器一旦装入初值，便马上开始计数，每当计数结束，便立即在输出端送出一个宽度等于一个时钟周期的负脉冲。如果在一次计数期间，装入了一个新的计数值。则在当前的计数结束，送出负脉冲后，马上以这个新的计数值开始计数。在GATE＝0时，禁止计数，这些均与方式2同，但这不是用GATE的上升沿来启动计数的。

方式5—硬件触发选通方式

当采用该方式工作时，在GATE信号的上升沿启动计数器开始计数，输出一直保持高电平，当计数结束时，输出一个宽度等于时钟周期的负脉冲。在此种方式下，GATE是高电平或低电平都不再影响计数器工作。但计数操作可用GATE信号的上升沿重新触发，便又从原来的初值开始计数，计数期间，输出又一直保持高电平。

在上述六种工作方式中，GATE信号均起作用，现将GATE信号的作用列于表中：

GATE	低电平或下降沿	上升沿	高电平
方式0	禁止计数	无作用	允许计数
方式1	无作用	启动计数， 下一时钟后输出变低	无作用
方式2	禁止计数并输出为高	启动计数	允许计数
方式3	禁止计数并输出为高	启动计数	允许计数
方式4	禁止计数	无作用	允许计数
方式5	无作用	启动计数	无作用

对8254的读写操作

写操作

由于每个计数器均有相应的口地址，方式控制字寄存器也有口地址，因而可用I/O写指令写入控制字和给计数器赋初值。当对8254编程写操作时，不一定按计数器0、1、2的顺序进行，计数器初值的装入，也不必按顺序或跟在写入该计数器控制字之后，但装入的计数器初值一定要按控制字RL0、RL1规定的顺序形式装入，如RL1＝RL0＝1，则一定要装入两个字节的数，而且必须先装低字节，然后再装高字节。

读操作

对8254进行读出计数值的操作可有两种方法，第一种是使计数器停止计数后，进行读操作，这可用GATE信号或外部硬件逻辑停止计数器工作，然后用I/O读指令进行读出，在读出时用写入控制字D4和D5是0或1来控制，若为11，则读数，先读出的是低8位，后读出的是高8 位，只有完成两次读后，才能进行别的操作，否则出错，若不是11，则读一次就可以了。

另一种读数方法是在计数过程中读，读时并不影响当时正在进行的计数，这时读出的是当前的计数值。这种读分两步进行：

第一步，写入专用字到控制字寄存器。其格式如下：

SC1　SC0　0　0　×　×　×　×

其中SC1、SC0与方式控制字中的规定一样，它们的组合，决定要选的计数器号，实际上这个控制字就是前面所说明的控制字的特殊形式（RL0＝RL1＝0）。该控制字实际完成将选定的计数器的当前计数值锁存到锁存器中，然后再用两条IN指令，就可以把锁存的值取出。

注意：当计数器的初值写入计数器后，计数器未开始计数前，写入的计数值，将不能读出。只有当计数器开始计数后，方能读出计数值。

7.2 8254 在HY—8130中的应用

在HY—8130板上，装有一片可编程定时/计数器8254，其中定时/计数器 1 和定时/计数器 2 已在硬件上级联起来，作为AD采样数据长度计数器。定时/计数器0使用 8MHz 的时钟，分频后做为A/D的启动信号。

注意：8254的每个定时/计数器在所有操作方式下，均不能设置初值为“1”，否则定时/计数器将停止计数及计数输出。

AD采样数据的计数长度可按如下公式计算：

$$L=[(HD2\times 256+LD2)\times(HD1\times 256+LD1)]/1024$$

其中：L为AD采样数据的计数长度，单位：K（1K=1024字节）

HD1为8254定时/计数器1初值的高字节，范围：0～255

LD1为8254定时/计数器1初值的低字节，范围：0～255

HD2为8254定时/计数器2初值的高字节，范围：0～255
LD2为8254定时/计数器2初值的低字节，范围：0～255

AD采样频率可按如下公式计算：

$$f = 8MHz / (HD0 \times 256 + LD0)$$

其中：f 为AD采样频率，单位：MHz

HD0为8254定时/计数器0初值的高字节，范围：0～255

LD0为8254定时/计数器0初值的低字节，范围：0～255

HY—8130板上的定时/计数器8254，是作为A/D板内部使用的定时/计数器，为了保证A/D板能够正常工作，定时/计数器0和定时/计数器1必须使用工作方式2，定时/计数器2必须使用工作方式0。

HY—8130板根据存储器容量不同有A型：128K，B型：512K两种。在软件设定AD采样数据长度计数器时，应根据所用存储器容量设定计数器长度。使用提前触发或滞后触发工作方式时应以此容量为常数，减去提前量或加上滞后量来作为计数器长度，提前、滞后量最大等于此容量，最小等于零。其它工作方式计数器长度应小于或等于该常数。

下面程序是用 1MHz 的采样频率，采集128K 数据（AD采样数据的计数长度＝128K）时8254三个定时/计数器的设置方法。

```
outp(baseAddr+14,0x74);          /*设置8254定时/计数器1控制字为方式2*/
outp(baseAddr+10,16);            /*写8254定时/计数器1的计数初值低字节*/
outp(baseAddr+10,0);             /*写8254定时/计数器1的计数初值高字节*/
outp(baseAddr+14,0xb0);          /*设置8254定时/计数器2控制字为方式0*/
outp(baseAddr+12,0);             /*写8254定时/计数器2的计数初值低字节*/
outp(baseAddr+12,32);            /*写8254定时/计数器2的计数初值高字节*/
                                 /*[(32×256+0)×(0×256+16)]/1024=128k*/
outp(baseAddr+14,0x34);          /*设置8254定时/计数器0控制字为方式2*/
outp(baseAddr+8,8);              /*写8254定时/计数器0的计数初值低字节*/
outp(baseAddr+8,0);              /*写8254定时/计数器0的计数初值高字节*/
                                 /*  8MHz /(0×256+8)＝1MHz  */
```

7.3 AD板触发方式

HY—8130板支持多种触发AD板工作的方式。触发方式是指由程序指令、外部模入被测信号或外部TTL电平脉冲使A/D板内部电路按初始化设定的参数开始工作的过程。应注意，触发方式不同于A/D启动方式。A/D启动方式是指板上逻辑电路启动A/D芯片工作的方式。A/D启动信号来自8254 定时/计数器 0 的输出和外部时钟脉冲的输入。

下面分别介绍每种触发方式的编程步骤，具体编程实例，请参阅7.6节。

7.3.1 软件触发

软件触发即用程序指令启动AD板，按初始化设定的模入通道范围、AD采样数据长度、采样频率等参数开始采集数据，并把采样结果存入板上存储器中。

软件触发AD板编程步骤如下：

1．板基地址＋0 写；设定AD采样模拟输入通道范围。

2．板基地址＋4 写；写板控制字，设定同步多采保、中断、板上存储 器工作方式、阈值电平工作方式等、关闭AD采样数据长度计数器、外部时钟脉冲控制、阈值电平允许控制寄存器；

3．写8254定时/计数器的控制字；写定时/计数器1、定时/计数器2的数据，设定AD采样数据长度计数器；HY—8130A型采样数据长度不超过128K；B型不超过512K。设置方法参见7.2节。

4．板基地址＋4 写；K0＝1：允许采样长度计数器计数。

5．板基地址＋8 写；设定AD采样频率；设置方法参见7.2节。

6．板基地址＋4 写；K8＝1：存储器地址复位。

7．板基地址＋6 读；由软件触发AD板开始采集数据。

8．板基地址＋4 读；查询数据采集结束状态。如使用中断方式则 等待中断申请信号。

9．板基地址＋4 写；K3＝0：允许从板上存储器中读AD转换结　　果。K8＝1：存储器地址复位。

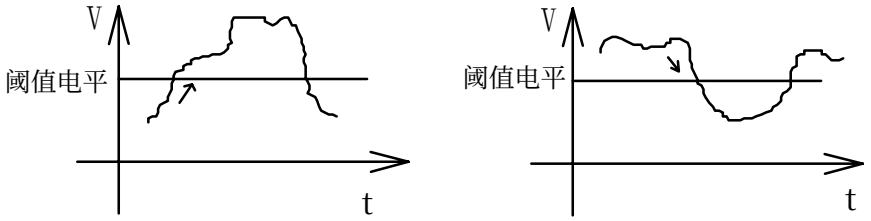
10．板基地址＋2 读；读AD转换结果。

11．板基地址＋0 读；使存储器地址＋1。

12．反复进行10、11步骤。直到板上存储器中的全部数据传送到　主机为止。

7.3.2 提前触发和滞后触发

HY—8130 板提前触发和滞后触发工作方式，适用于观测触发时刻前后一段时间内的各通道信号变化情况。触发条件一但满足，立刻产生触发信号，启动内部电路，按照程序设定的模入通道范围、AD采样数据长度计数器、采样频率等参数开始采集数据，同时把采样结果放入板上存储器中。



当外部模入信号第一次交越阈值触发电平时，产生触发信号，AD采样数据长度计数器开始计数。

提前触发工作方式将模入通道0或通道4的模拟量输入信号电平（由XF3跳线选择），作为触发信号电平。用于观测触发信号电平第一次以上升沿或下降沿交越设定的采样触发阈值电平时刻之前，一段时间（提前时间TA）以后，各通道信号变化情况（参见图7.3.3-1），提前量可由程序设定的AD采样数据长度计数器计算公式导出。请参阅本节导出计算公式。

滞后触发工作方式将模入通道0（CH0）或通道4（CH4）的模拟量输入信号电平（由XF3跳线选择），作为触发信号电平。用于观测触发信号电平第一次以上升沿或下降沿交越设定的采样触发阈值电平时刻之后，一段时间（滞后时间TB）以后，各通道信号变化的情况（参见图7.3.3-2），滞后量可由程序设定的AD采样数据长度计数器计算公式导出。

注意：如果用上升沿触发AD板工作，在触发模入信号的有效上升沿到达之前，设定的阈值触发电平应始终高于选定的模入信号电平。用下降沿触发AD板工作，设定的阈值触发电平应始终低于选定的模入信号电平。

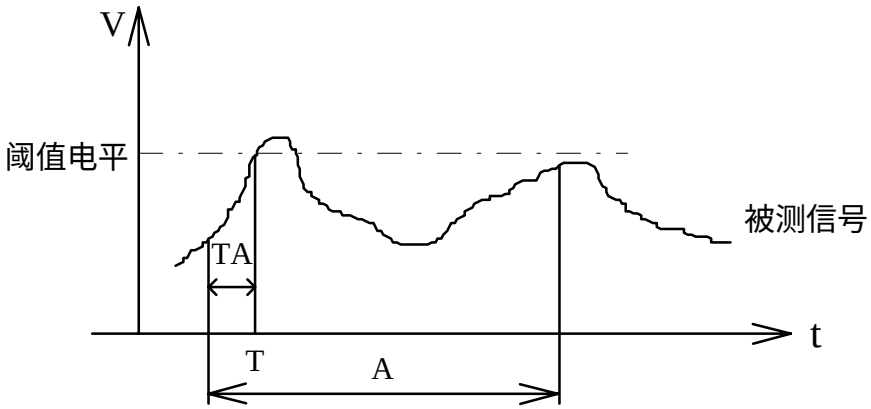


图7.3.3-1 提前触发方式，存储器中采样数据与时间的关系图

图7.3.3-1中：T：触发时刻。

TA：提前时间，可根据需要选择。

A：提前触发方式记录信号长度，根据存储器配置为128K或512K而定。

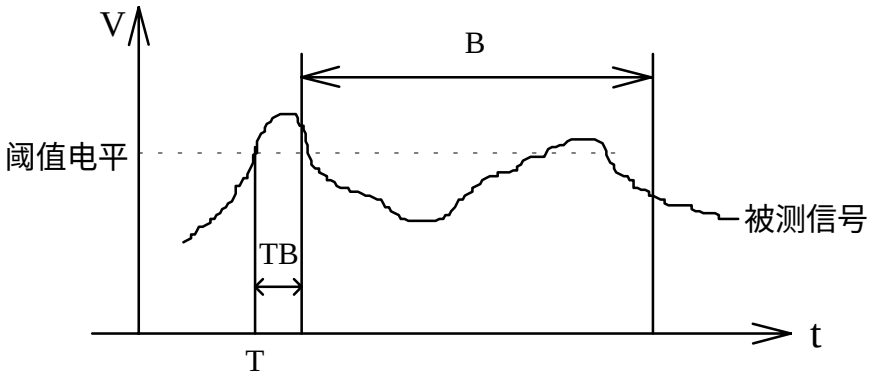


图7.3.3-2 滞后触发方式，存储器中采样数据与时间的关系图

图7.3.3-2中：T：触发时刻。

TB：滞后时间，可根据需要选择。

B：滞后触发方式记录信号长度，根据存储器配置为128K或512K而定。

提前（滞后）量与时间的对应关系可按如下公式计算：（同步多采保不工作时）

$$tzl = (t \times f \times s) / 1024$$

其中： *tzl*为提前（滞后）量，单位：K（1K=1024字）

t 为提前（滞后）时间，单位：S（秒）

f 为采样频率，单位：Hz

s 为通道循环数，范围：1～8

提前（滞后）量与时间的对应关系可按如下公式计算：（同步多采保工作时）

$$tzl = [t \times f \times s - 1] / 1024$$

其中： *tzl*为提前（滞后）量，单位：K（1K=1024字）

t 为提前（滞后）时间，单位：S（秒）

f 为采样频率，单位：MHz

s 为通道循环数，范围：1～8

提前触发时AD采样数据长度计数器赋值按如下公式计算：

$$L = rl - tzl$$

其中：*L* 为计数器长度，单位：K（1K=1024字）

rl 为板上存储器容量。（A型：128K；B型：512K）

tzl 为提前量，单位：K（1K=1024字）

滞后触发时AD采样数据长度计数器赋值按如下公式计算：

$$L = rl + tzl$$

其中：*L* 为计数器长度，单位：K（1K=1024字）

rl 为板上存储器容量。（A型：128K；B型：512K）

tzl 为滞后量，单位：K（1K=1024字）

HY—8130板的AD采样数据长度计数器赋值计算公式，由板硬件结构导出。用户只要按照公式计算，即可采到相应时间段的数据。

下面，针对AD采样数据长度计数器的设置方法举两个例子。请参照执行。

例1：观测从触发条件满足以前4ms 时刻开始，8 个模入通道被测信号的变化过程。应选用提前触发方式。（实际编程请参阅7.6.2节）

设：板上存储器容量为128K，AD采样频率为1MHz

当同步多采保不工作时提前量为：

提前量＝提前时间×采样频率×通道循环数

$$tzl = (t \times f \times s) / 1024$$

＝4ms×1MHz×8＝4×10^{-3}S×10^{6}/S×8/1024＝32K

经过计算得到提前量等于32K。

同步多采保不工作时提前量AD采样数据长度计数器设置为：

计数器长度＝板上存储器容量－提前量

$$L = rl - tzl$$

＝128K－32K＝96K

当同步多采保工作时提前量为：

提前量＝提前时间×采样频率×（通道循环数－1）

$$tzl = t \times f \times (s-1)/1024$$

＝4×10^{-3}S×10^{6}/S×（8－1）/1024＝28K

经过计算得到提前量等于28K。

同步多采保工作时AD采样数据长度计数器设置为：

计数器长度＝板上存储器容量－提前量

$$L = rl - tzl$$

＝128K－28K＝100K

采样完成后从板上存储器连续读取的128K（A型：128K。B型：512K）数据，即为包括触发信号来到前4mS时的全部有效数据。8254定时/计数器1 和定时/计数器2 应做如下初始化：

```
outp(baseAdd+14,0x74);          /*设置8254定时/计数器1控制字*/
outp(baseAdd+10,8);             /*写8254定时/计数器1的计数初值低字节*/
outp(baseAdd+10,0x00);          /*写8254定时/计数器1的计数初值高字节*/
outp(baseAdd+14,0xb0);          /*设置8254定时/计数器2控制字*/
outp(baseAdd+12,0x00);          /*写8254定时/计数器2的计数初值低字节*/
outp(baseAdd+12,50);            /*写8254定时/计数器2的计数初值高字节*/
                                /*[(50×256+0)×(0×256+8)]/1024＝100K*/
```

例2：观测触发条件满足4ms 时间以后，8 个模入通道被测信号的变化过程。应选用滞后触发方式。（实际编程请参阅7.6.3节）

设：板上存储器容量　　N＝128K（2^{13}＝131072）

AD采样频率＝1MHz

同步多采保不工作时滞后量为：

滞后量 ＝滞后时间×采样频率×通道循环数/1024

$$=(t \times f \times s)/1024$$

＝4ms×1MHz×8＝4×10^{-3}S×10^{6} /S×8/1024＝32K

计数器长度 ＝存储器容量＋滞后量

$$=rl+tzl$$

＝128K＋32K＝160K

同步多采保工作时滞后量为：

滞后量 ＝滞后时间×采样频率×（通道循环数－1）/1024

$$=t \times f \times (s-1)/1024$$

＝4×10^{-3}S×10^{6}/S×（8－1）/1024＝28K

计数器长度 ＝存储器容量＋滞后量

$$=rl+tzl$$

＝128K＋28K＝156K

采样完成后从板上存储器连续读取的128K（A型：128K。B型：512K）数据，即为从触发信号来到后4mS时间以后的全部有效数据。8254 定时/计数器1 和定时/计数器2 应做如下初始化操作：

```
outp(baseAdd+14,0x74);          /*设置8254定时/计数器1控制字*/
outp(baseAdd+10,8);             /*写8254定时/计数器1的计数初值低字节*/
outp(baseAdd+10,0x00);          /*写8254定时/计数器1的计数初值高字节*/
outp(baseAdd+14,0xb0);          /*设置8254定时/计数器2控制字*/
```

```
outp(baseAdd+12,0x00);          /*写8254定时/计数器2的计数初值低字节*/
outp(baseAdd+12,78);            /*写8254定时/计数器2的计数初值高字节*/
                                /*[(78×256+0)×(0×256+8)]/1024＝156K */
```

定时/计数器 8254 的使用细则请参阅本说明书7.2 节 。

提前触发和滞后触发AD板初始化编程步骤如下：

1．板基地址＋2 写；设定阈值触发电平。

2．板基地址＋0 写；设定AD采样模拟输入通道范围。

3．板基地址＋4 写；设定同步多采保、中断、AD采样数据长度计数器，板上存储器控制等，设定阈值电平为外部模入信号上升沿触发有效，或是下降沿触发有效，打开阈值电平采样工作允许寄存器、关闭外部时钟脉冲输入。

4．写定时/计数器8254的控制字，写定时/计数器1、定时/计数器2 的数据，设定AD采样数据长度计数器；设置方法参见7.2节。

5．板基地址＋4 写；K0＝1：允许计数。

6．板基地址＋8 写；设定AD采样频率；设置方法参见7.2节。

7．板基地址＋4 读；查询数据采集结束状态。如使用中断方式则 等待中断申请信号。

8．板基地址＋4 写；K3＝0：允许从板上存储器中读AD转换结 果。

9．板基地址＋2 读；读AD转换结果。

10．板基地址＋0 读；使存储器地址＋1。

11．反复进行9、10步骤128K次（A型板）或512K次（B型板），将全部AD采集结果读到主机内存。

由AD板向计算机传送数据之前，存储器地址指针所对应的地址，对提前触发信号来说，是触发信号来到时刻之前若干时间（由程序设定），所对应的数据地址。对滞后触发信号来说，是触发信号来到时刻之后若干时间（由程序设定），所对应的数据地址。从这一地址开始，连续读取的128K（A型：128K。B型：512K）数据，为有效数据。提前或滞后量的计算方法参见7.6节编程实例。

7.3.3 外触发工作方式

HY—8130板的外触发工作方式，适合于用外触发脉冲作采样同步信号的应用场合，每来一个外触发脉冲，AD板在设定的采样通道范围内对各模入通道循环采样一遍。外触发脉冲周期应大于设定模入通道循环采样一遍所用时间。外触发脉冲周期可按下面公式计算：

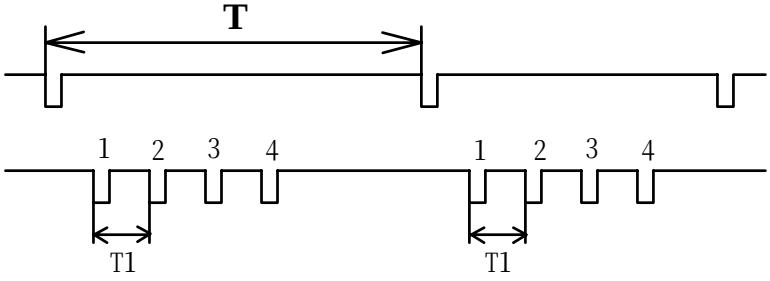
$$T > (N+1)T1$$

其中： T为外触发脉冲周期，单位：S（秒）

T1为AD采样周期，单位：S（秒）

N为采样通道数（末通道—首通道＋1），范围：1～8

例如下图：



T＞（4＋1）×T1 (N＝4)

外触发脉冲应为标准TTL电平负脉冲。HY—8130板上外触发脉冲输入端接有10K上拉电阻。外触发信号经过板上单稳整形后，供给板上控制逻辑使用。所以，对外触发信号的脉宽没有严格要求。通道间采样频率由编程决定。用外触发脉冲来同步板上可编程定时计数器8254的输出，这样会使采样通过率大约下降百分之二十，最高仅到800KHz，请用户注意。

外触发信号地线应接入XS1插座上的MGND地线端（参阅4.2节）。

外触发方式编程步骤如下：

1．板基地址＋0 写；设定AD采样模入通道范围。

2．板基地址＋4 写；设定同步多采保、中断、AD采样数据缓冲存储器、阈值电平工作方式等、关闭AD采样数据长度计数器、阈值电平允许控制寄存器；禁止外触发工作。

3．写定时/计数器8254的控制字，写定时/计数器1、定时/计数器2的数据，设定AD采样计数器长度；HY—8130A型采样数据长度不超过128K；HY—8130B型不超过512K。设置方法参见7.2节。

4．板基地址＋4 写；K0＝1：允许计数。

5．设定AD采样频率；设置方法参见7.2节。

6．板基地址＋4 写；K8＝1：存储器地址复位。K6＝1：允许外部　　脉冲启动AD工作。

7．板基地址＋4 读；查询数据采集结束状态。如使用中断方式则　等待中断申请信号。

8．板基地址＋4 写；K3＝0：允许从存储器中读AD转换结果。

K8＝1：存储器地址复位。

K6＝0：禁止外部脉冲启动AD工作。

K9＝1：清AD采集结束标志寄存器。

9．板基地址＋2 读；读AD转换结果。

10．板基地址＋0 读；使存储器地址＋1。

11．反复进行9、10步骤128K次（A型板）或512K次（B型板），将全部AD采集结果读到主机内存。将全部采集结果读进主机存储器。

7.4 外部时钟启动AD采样工作方式

HY—8130板的外部时钟启动AD采样工作方式，适合于用外部输入的TTL 电平负脉冲作为AD芯片启动信号的应用场合，每来一个外部时钟脉冲，AD芯片启动转换 1 次，模入采样通道在设定的通道循环范围内自动切换至下一通道。外部输入时钟信号周期（T）不应小于1μs的AD芯片转换时间。其中，负电平有效时间不应大于500ns。外部时钟脉冲逻辑电平为：CMOS 电平或TTL 电平兼容负脉冲。HY—8130板上外部时钟脉冲输入端接有10K上拉电阻。外部时钟信号经过板上施密特门整形后，供给板上控制逻辑使用。

外部时钟地线应接入XS1插座上的MGND地线端（参阅4.2节）。

外部时钟启动AD采样，AD板初始化编程步骤如下：

1．板基地址＋0 写，设定AD采样模拟输入通道范围。

2．板基地址＋4 写，设定同步多采保、中断、板上存储器工作方式、阈值电平工作方式等、关闭AD采样数据长度计数器、阈值电平允许控制寄存器；禁止外部时钟启动AD采样工作。

3．写定时/计数器8254的控制字，写定时/计数器1、定时/计数器2的数据，设定AD采样计数器长度；HY—8130A型采样计数器长度不超过128K；HY—8130B型不超过512K。设置方法参见7.2节。

4．板基地址＋4 写，K0＝1：允许计数。

K8＝1：存储器地址复位。

K6＝1：允许外部脉冲启动AD工作。

5．板基地址＋4 读，查询数据采集结束状态。如使用中断方式则　等待中断申请信号。

6．板基地址＋4 写，K3＝0：允许从板上存储器中读AD转换结　　果。K8＝1：存储器地址复位。

7．板基地址＋2 读AD转换结果。

8．板基地址＋0 读，使存储器地址＋1。

9．反复进行7、8步骤，次数等于设定好的AD采样数据长度。将　全部采集结果读进主机存储器。

7.5 多块HY—8130板同步工作方式

HY—8130板上设有八路高速AD模拟信号输入通道，当需要更多通道进行高速数据采集时，可将多块HY—8130板插在同一台微机内，用设定不同的板基地址的方法，实现主机对不同A/D板的控制。可选定其中一块作为主板，其它作为从板，由主板的XS1插座的11脚输出AD转换启动信号，供从板由各自XS1插座的25脚（外时钟输入端）引入，作为从板的AD启动信号。从而构成同步多路高速数据采集系统。在多块HY—8130板组成的数据采集系统中，主板应工作在软件触发方式（参见7.3.1节），从板应工作在外部时钟启动AD采样工作方式（参见7.4节）。

7.6 编程实例

用户可以使用汇编、C、PASCAL等语言编写HY—8130板工作的程序。

首先设置好板基地址及相关的选择开关和相关的跳线。

用户编程如有困难，请购买本公司的驱动函数库。

注意：

（1）由于器件原因，每次采样的前 8 个数据有可能是不准确数　据。用户在编程时，可舍去，或复盖。

（2）Turbo　C 和 Quick　C，在个别语句上有如下区别：

Turbo　C	Quick　C
outport	outpw
inport	inpw

（3）以下编程实例，是以A型板（板上存储器容量：128K）为例进行编程的。如果使用B型板（板上存储器容量：512K），可按7.2节A/D采样长度计数器的计算公式设置计数长度。

7.6.1 软件触发工作方式编程实例

下面例程为使用软件触发方式工作的采样过程

```
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>
unsigned far ADbuffer[4000][8];
unsigned far buffer[80];
unsigned baseAddr=0x280;
void main(void)
{
    int i,j,g;
    int aa=0;                              /* 开始对 HY—8130 A/D 板初始化 */
    outport   (baseAddr,0x0070);           /* 设置硬件自动切换通道循环范围 0～7 */
    outport (baseAddr+4,0x0108);           /* 板上存储器地址清零；关多采保；关外触发；禁止外部阈值电平触发；AD转换结果
    写入板上存储器；关中断；关中断允许；关采样长度计数器 */
    outp (baseAddr+14,0x74);               /* 设置8254 定时/计数器1 为工作方式2 */
    outp (baseAddr+10,4);                  /* 设置A/D采样长度计数器的低16 位 */
    outp (baseAddr+10,0);
    outp (baseAddr+14,0xb0);               /* 设置8254 定时/计数器2 为工作方式 0 */
    outp (baseAddr+12,0);
    outp (baseAddr+12,128);                /* 设置A/D采样长度计数器的高16 位 */
                                  /* A/D采集长度等于(128*256+0)*(0*256+4)=128k */
    outport   (baseAddr+4,0x0109);         /* 打开A/D采样长度计数器 */
    outp (baseAddr+14,0x34);               /* 设置8254 定时/计数器0 为工作方式 2 */
    outp (baseAddr+8,8);
    outp (baseAddr+8,0);                       /* 设置A/D采样频率为： 1MHz */
    inp(baseAddr+6);                           /* 按初始化设置内容开始工作 */
```

```
                                        /* HY—8130 A/D 板初始化完成 */
    do {                                /* 等待 A/D 采样完成  */
        aa=(inp(baseAddr+4) & 0x0001);  /* A/D 采样完成，读0x284口d0=1 */
    } while (aa==0);
    outport (baseAddr+4,0x0100);
                /* 设定由程序从板上存储器中读AD转换结果；存储器地址复位 */
    for (j=0;j<8;j++){                  /* 舍去前8个不稳定数据 */
        inp (baseAddr+0) ;              /* 存储器地址+1 */
    }
    for (j=0;j<8;j++){                  /* 查找模入首通道 */
        buffer[j]=inport(baseAddr+2)&0x8000;
        inp (baseAddr+0) ;              /* 存储器地址+1  */
        if (buffer[j]==0)   break;
    }
    clrscr();
    for ( i=0; i<3984; i++){
        if (i%20==0){
          printf("\n\npress Empty key display A/D sample data     press   ESC key Exit\n");
          g=getch();
          if (g==27)   break;
          clrscr();
          printf("ch0\t ch1\t ch2\t ch3\t ch4\t ch5\t ch6\t ch7\t\n");
        }
        printf("\n");
        for (j=0;j<8;j++){              /* 从板上存储器向主机传送AD转换结果 */
            ADbuffer[i][j]=inport(baseAddr+2)&0xfff;
            printf("%+2.3fv \t",(float)((int)ADbuffer[i][j])*0.0024492-5 .016);
                                        /* 显示±5.016v模入范围的采样结果 */
            inp (baseAddr+0) ;          /* 存储器地址+1  */
        }
    }
}
```

7.6.2 提前触发工作方式编程实例

下面例程为观测触发条件满足之前4ms时间以后，8个模入通道被测信号的变化过程。

```
/*下面例程为使用提前触发方式工作的采样程序*/
#include <stdlib.h>
#include <alloc.h>
#include <stdio.h>
#include <conio.h>
unsigned far ADbuffer[32767];
unsigned baseAddr=0x280;
void main(void)
{
    unsigned long   a,b,aa=0,g,h,i,j,bz,n,z; /* 开始对HY—8130A/D板初始化 */
    clrscr();
    outport   (baseAddr+2,0x0a00);      /*设置提前采样触发阈值电平等于+1.25V*/
    outport   (baseAddr,0x0070);        /* 设置硬件自动切换通道循环范围0～7 */
    outport   (baseAddr+4,0x0038);      /* 关多采保；关外触发；允许外部阈值电平触发；外部模入信号上升沿交越阈值电平
    触发有效；AD转换结果写入板上存储器关中断；关中断允许；关采样长度计数器  */
    outp (baseAddr+14,0x74);            /* 设置8254 定时/计数器1 为工作方式2 */
    outp (baseAddr+10,3);
    outp (baseAddr+10,0);               /* 设置A/D采样长度计数器的低16 位 */
    outp (baseAddr+14,0xb0);            /* 设置8254 定时/计数器2 为工作方式 0 */
    outp (baseAddr+12,0);
    outp (baseAddr+12,128);             /* 设置A/D采样长度计数器的高16 位 */
                                /* A/D采集长度等于(128*256+0)*(0*256+3)=96k */
    outp (baseAddr+14,0x34);            /* 设置8254 定时/计数器0 为工作方式 2 */
    outp (baseAddr+8,8);
    outp (baseAddr+8,0);                /* 设置A/D采样频率为： 1MHz */
    inp(baseAddr+6);                    /* HY—8130 A/D 板初始化完成 */
    printf("No Trigger Singel,   Please Wait. \n");
    do{                                 /* 等待 A/D 采样完成  */
```

```
        aa=(inp(baseAddr+4) & 0x0001);  /* A/D采样完成，读0x284口d0=1 */
    }while (aa==0);
    outport (baseAddr+4,0x0030);          /*  由程序从板上存储器中读AD转换结果  */
    for (j=0;j<32767;j++){                /*  寻找触发点  */
        inp (baseAddr+0);                 /*  存储器地址+1 */
    }
    clrscr();
    for (i=0; i<4000; i++){
        if (i%23==0){
          printf("\n\npress Empty key display A/D sample data    press    ESC key Exit\n");
          g=getch();
          if (g==27)    break;
          clrscr();
        }
        printf("\n");
        for (j=0;j<8;j++){                /*  从板上存储器向主机传送AD转换结果  */
            ADbuffer[j]=inport(baseAddr+2)&0xfff;
            printf("%+2.3fv \t",(float)((int)ADbuffer[j])*0.0024492-5.016 );
                    /*  显示±5.016v模入范围触发时刻之前4ms以后的采样结果  */
            inp (baseAddr+0);             /*  存储器地址+1 */
        }
    }
}
```

7.6.3 滞后触发工作方式编程实例

下面例程为观测触发条件满足4ms时间之后，8个模入通道被测信号的变化过程。

```
/*下面例程为使用滞后触发方式工作的采样程序*/
#include <stdlib.h>
#include <alloc.h>
#include <stdio.h>
#include <conio.h>
unsigned far ADbuffer[32767];
unsigned baseAddr=0x280;
void main(void)
{
    unsigned long    a,b,aa=0,g,h,i,j,bz,n,z;      /*  开始对  HY—8130 A/D  板初始化    */
    clrscr();
    outport    (baseAddr+2,0x0a00);       /*  设置提前采样触发阈值电平等于+1.25V */
    outport    (baseAddr,0x0070);         /*  设置硬件自动切换通道循环范围  0～7 */
    outport    (baseAddr+4,0x0038);       /*  关多采保；关外触发；允许外部阈值电平触发；外部模入信号上升沿交越阈值电平触
    发有效；AD转换结果写入板上存储器关中断；关中断允许；关采样长度计数器  */
    outp (baseAddr+14,0x74);              /*  设置8254  定时/计数器1  为工作方式2 */
    outp (baseAddr+10,5);
    outp (baseAddr+10,0);                 /*  设置A/D采样长度计数器的低16  位  */
    outp (baseAddr+14,0xb0);              /*  设置8254  定时/计数器2  为工作方式  0 */
    outp (baseAddr+12,0);
    outp (baseAddr+12,128);               /*  设置A/D采样长度计数器的高16  位  */
                                  /* A/D采集长度等于(128*256+0)*(0*256+5)=160k */
    outp (baseAddr+14,0x34);              /*  设置8254  定时/计数器0  为工作方式  2 */
    outp (baseAddr+8,8);
    outp (baseAddr+8,0);                  /*  设置A/D采样频率为：  1MHz */
    inp(baseAddr+6);                      /* HY—8130 A/D  板初始化完成  */
    printf("No Trigger Singel,    Please Wait. \n");
    do{                                       /*  等待  A/D  采样完成    */
        aa=(inp(baseAddr+4) & 0x0001);        /* A/D  采样完成，读0x284口d0=1 */
    }while (aa==0);
    outport (baseAddr+4,0x0030);          /*  由程序从板上存储器中读AD转换结果  */
    clrscr();
    for(i=0; i<4000; i++){
        if (i%23==0){
          printf("\n\npress Empty key display A/D sample data    press    ESC key Exit\n");
          g=getch();
          if (g==27)    break;
```

```
            clrscr();
        }
        printf("\n");
        for(j=0;j<8;j++){                    /*  从板上存储器向主机传送AD转换结果  */
            ADbuffer[j]=inport(baseAddr+2)&0xfff;
            printf("%+2.3fv \t",(float)((int)ADbuffer[j])*0.0024492-5.016 );
                                /*  显示±5.016v模入范围触发时刻4ms以后的采样结果  */
            inp (baseAddr+0) ;                /*  存储器地址+1    */
        }
    }
}
```

7.6.4 中断方式采样编程实例

```
/*下面例程为使用中断方式的采样程序*/
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>
unsigned _far ADbuffer[4000][8];
unsigned baseAddr=0x280,T1,T2;
unsigned DoneFlag=0;
void interrupt far(*oldIntSever)(void);          /*  中断函数说明  */
void ADaquire(void);

void interrupt myIntPgm(void)                    /*  中断函数  */
{
    int i,j,a;
    outport(baseAddr+4,0x104);        /*  板上存储器地址清零；由程序从板上采样缓冲存储器中读AD转换结果；关多采保；关外触发；禁止外部阈值电平触发；开中断允许；关中断申请；关采样长度计数器  */
    for (i=0; i<4000 ; i++ ){
        for (j=0;j<8;j++){                   /*  从板上存储器向主机传送AD转换结果  */
            ADbuffer[i][j]=inport(baseAddr+2)&0xfff;
            inp (baseAddr+0);                /*  存储器地址+1 */
        }
    }
    outport (baseAddr+4,0x108);              /*  清中断  */
    DoneFlag=1;                              /*  设置中断标志  Doneflag=1 */
    outp(0xa0,0x20);                         /*  通知计算机  2# 8259  中断结束  */
    outp(0x20,0x20);                         /*  通知计算机  1# 8259  中断结束  */
}
void ADaquire()
{
    outport(baseAddr+4,0x10d);        /*  板上存储器地址清零；AD转换结果写入板上存储器；关多采保；关外触发；禁止外部阈值电平触发；关中断允许；开采样长度计数器  */
    enable();                                /*  主机中断允许  */
    inp(baseAddr+6);                         /*  按初始化设置内容开始工作  */
    while(DoneFlag==0);                      /*  等待中断来到  */
}

void main(void)
{
    int mode,inkey;
    int i,j,aa,bz,g;
    FILE *outfp;
    disable();                               /*  关中断  */
    clrscr();
    oldIntSever=getvect(0x72);               /*  保存原来的中断向量  */
    setvect(0x72,myIntPgm);                  /*  修改计算机中断屏蔽字  */
    outp(0xa1,inp(0xa1)&0xfb);               /*  设置计算机  8259  中断向量：IRQ10 */
                                             /*  开始对  HY—8130 A/D  板初始化  */
    outport    (baseAddr,0x0070);            /*  设置硬件自动切换通道循环范围  0～7 */
    outport    (baseAddr+4,0x0108);          /*  板上存储器地址清零；AD转换结果写入板上存储器；关多采保；关外触发；禁止外部阈值电平触发；开中断允许；关中断申请；关采样长度计数器  */
```

```
    outp (baseAddr+14,0x74);              /* 设置8254 定时/计数器1 为工作方式2 */
    outp (baseAddr+10,4);
    outp (baseAddr+10,0);                 /* 设置A/D采样长度计数器的低16 位 */
    outp (baseAddr+14,0xb0);              /* 设置8254 定时/计数器2 为工作方式 0 */
    outp (baseAddr+12,0);
    outp (baseAddr+12,128);               /* 设置A/D采样长度计数器的高16 位 */
                                /* A/D采集长度等于(128*256+0)*(0*256+4)=128k */
    outport   (baseAddr+4,0x0109);        /*打开A/D采样长度计数器 */
    outp (baseAddr+14,0x34);              /* 设置8254 定时/计数器0 为工作方式 2 */
    outp (baseAddr+8,8);                  /* 设置A/D采样频率为： 1MHz */
    outp (baseAddr+8,0);
    inp(baseAddr+6);                      /* HY—8130 A/D 板初始化完成 */
    do{
        DoneFlag=0;                                 /* 清中断标志 */
        ADaquire();                                 /* 调用采样模块 */
        for(j=0;j<8;j++){                           /* 舍去前8个不稳定数据 */
            inp (baseAddr+0);                       /* 存储器地址+1 */
        }
        for(j=0;j<8;j++){                           /* 舍去前8个不稳定数据 */
            bz=inport(baseAddr+2)&0x8000;
            inp (baseAddr+0);                       /* 存储器地址+1   */
            if (bz==0)   break;
        }
        clrscr();
        for(i=0; i<4000; i++){
          if(i%20==0){
            printf("\n\npress Empty key display A/D sample data press ESC key Exit\n");
            g=getch();
            if (g==27)   break;
            clrscr();
            printf("ch0\t ch1\t ch2\t ch3\t ch4\t ch5\t ch6\t ch7\t\n");
          }
          printf("\n");
          for (j=0;j<8;j++){                        /* 从板上存储器向主机传送AD转换结果 */
              ADbuffer[i][j]=inport(baseAddr+2)&0xfff;
              printf("%+2.3fv \t",(float)((int)ADbuffer[i][j])*0.0024492-5.016 );
                                                    /* 显示±5.016v模入范围的采样结果 */
              inp (baseAddr+0);                     /* 存储器地址+1   */
          }
        }
        if (g==27)   break;
    }while((inkey&0xff)!='N' && (inkey&0xff)!='n');
    setvect(0x72,oldIntSever);                      /* 恢复原来的中断向量 */
}
```

7.6.5 外部时钟启动A/D采样编程实例

```
/*下面例程为使用外部时钟启动A/D采样的编程实例*/
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>
unsigned _far ADbuffer[4000][8];
unsigned baseAddr=0x280,T1,T2;
void main(void)
{
    int i,j,bz,g;
    int aa=0;                             /* 开始对 HY—8130 A/D 板初始化 */
    outport (baseAddr,0x0070);            /* 设置硬件自动切换通道循环范围 0～7 */
    outport (baseAddr+4,0x0108);          /* 板上存储器地址清零；关多采保；关外触发；禁止外部阈值电平触发；关中断；关
    中断允许；关采样长度计数器 */
    outp (baseAddr+14,0x74);              /* 设置8254 定时/计数器1 为工作方式2 */
    outp (baseAddr+10,4);
    outp (baseAddr+10,0);                 /* 设置A/D采样长度计数器的低16 位 */
```

```
    outp (baseAddr+14,0xb0);              /* 设置8254 定时/计数器2 为工作方式 0 */
    outp (baseAddr+12,0);
    outp (baseAddr+12,128);               /* 设置A/D采样长度计数器的高16 位 */
                                  /* A/D采集长度等于(128*256+0)*(0*256+4)=128k */
    inp(baseAddr+6);                      /* 按初始化设置内容开始工作 */
                                          /* HY—8130 A/D 板初始化完成 */
    outport (baseAddr+4,0x0149);          /* 允许外时钟 */
    do{                                   /* 等待 A/D 采样完成 */
        aa=(inp(baseAddr+4) & 0x0001);    /* A/D 采样完成，读0x284口d0=1 */
    } while (aa==0);
    outport (baseAddr+4,0x0100);          /* 板上存储器地址清零；设定由程序从板      上存储器中读AD转换结果；关外时钟
    */
    for(j=0;j<8;j++){                     /* 舍去前8个不稳定数据 */
        inp (baseAddr+0);                 /* 存储器地址+1 */
    }
    for(j=0;j<8;j++){                                 /* 舍去前8个不稳定数据 */
        bz=inport(baseAddr+2)&0x8000;
        inp (baseAddr+0);                             /* 存储器地址+1 */
        if (bz==0)    break;
    }
    clrscr();
    for( i=0; i<4000; i++){
        if(i%20==0){
          printf("\n\npress Empty key display A/D sample data      press    ESC key Exit\n");
          g=getch();
          if (g==27)    break;
          clrscr();
          printf("ch0\t ch1\t ch2\t ch3\t ch4\t ch5\t ch6\t ch7\t\n");
        }
        printf("\n");
        for (j=0;j<8;j++){                /* 从板上存储器向主机传送AD转换结果 */
            ADbuffer[i][j]=inport(baseAddr+2)&0xfff;
            printf("%+2.3fv \t",(float)((int)ADbuffer[i][j])*0.0024492-5.016 );
                                          /* 显示±5.016v模入范围的采样结果 */
            inp (baseAddr+0);             /* 存储器地址+1 */
        }
    }
}
```

7.6.6 外部触发编程实例

```
/*下面例程为使用外部TTL信号触发扫描A/D模入通道的编程实例*/
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>
unsigned _far ADbuffer[4000][8];
unsigned baseAddr=0x280,T1,T2;

void main(void)
{
    int i,j,bz,g;
    int aa=0;                             /* 开始对 HY—8130 A/D 板初始化  */
    outport (baseAddr,0x0070);            /* 设置硬件自动切换通道循环范围 0～7 */
    outp (baseAddr+14,0x74);              /* 设置8254 定时/计数器1 为工作方式2 */
    outp (baseAddr+10,4);
    outp (baseAddr+10,0);                 /* 设置A/D采样长度计数器的低16 位 */
    outp (baseAddr+14,0xb0);              /* 设置8254 定时/计数器2 为工作方式 0 */
    outp (baseAddr+12,0);
    outp (baseAddr+12,128);               /* 设置A/D采样长度计数器的高16 位 */
                                  /* A/D采集长度等于(128×256+0)×(0×256+4)=128k */
    outp (baseAddr+14,0x34);              /* 设置8254 定时/计数器0 为工作方式 2 */
    outp (baseAddr+8,8);
    outp (baseAddr+8,0);                  /* 设置A/D采样频率为：1M */
```

```
                                        /* HY—8130 A/D  板初始化完成  */
    outport (baseAddr+4,0x010b);        /*  允许外触发  */
                                        /* HY—8130 A/D  板初始化完成  */
    do{                                 /*  等待  A/D  采样完成    */
        aa=(inp(baseAddr+4) & 0x0001);  /* A/D  采样完成，读0x284口d0=1 */
    } while (aa==0);
    outport (baseAddr+4,0x0100);        /*  板上存储器地址清零；设定由程序从板       上存储器中读AD转换结果；关外触发
    */
    for (j=0;j<8;j++){                  /*  舍去前8个不稳定数据  */
        inp (baseAddr+0);               /*  存储器地址+1 */
    }
    for(j=0;j<8;j++){                           /*  舍去前8个不稳定数据  */
        bz=inport(baseAddr+2)&0x8000;
        inp (baseAddr+0);                       /*  存储器地址+1 */
        if (bz==0)    break;
    }
    clrscr();
    for(i=0; i<4000; i++){
        if (i%20==0){
          printf("\n\npress Empty key display A/D sample data      press    ESC key Exit\n");
          g=getch();
          if (g==27)    break;
          clrscr();
          printf("ch0\t ch1\t ch2\t ch3\t ch4\t ch5\t ch6\t ch7\t\n");
        }
        printf("\n");
        for(j=0;j<8;j++){               /*  从板上存储器向主机传送AD转换结果  */
            ADbuffer[i][j]=inport(baseAddr+2)&0xfff;
            printf("%+2.3fv \t",(float)((int)ADbuffer[i][j])*0.0024492-5.016 );
                                        /*  显示±5.016v模入范围的采样结果  */
            inp (baseAddr+0);           /*  存储器地址+1 */
        }
    }
}
```

八、应用注意事项

8.1 注意事项

在公司售出的产品中，用户将会找到这本说明书和HY—8130板，同时还有产品质保卡及产品测试程序软盘，产品测试程序的使用详见附录1，HY—8130板的驱动函数使用说明详见附录2。用户需要使用驱动函数库时，请与本公司销售部门联系购买。产品质保卡请用户务必妥善保存，当该产品出现问题需要维修时，请用户将产品质保卡同产品一起，寄回本公司，以便我们能尽快的帮用户解决问题。

在使用HY—8130板时，应注意以下问题：

① HY—8130板正面的IC芯片不要用手去摸，防止芯片受到静电的危害。

② 用户务必注意电源的开关顺序，使用时要求先开主机电源，后开信号源的电源。先关信号源的电源，后关主机电源。

③ HY—8130板上插座、跳线和控制字中保留位，用户不要使用。

④ 外时钟信号应为标准TTL时钟信号，如果存在干扰会直接影响采集数据的正确性。

九.校准

9.1 校准准备

HY—8130板出厂时已经校准，只有当用户使用一段时间后，或者用户认为需要时才做校准。

①准备一块5位半精度以上数字电压表，模拟输入范围为
－5.016V～＋5.016V（出厂状态）单端模拟输入方式。

②安装好HY—8130，打开主机电源，预热15分种。

9.2 A/D校准

①选择模拟输入的任一通道，比如CH0通道，将模拟输入通道CH0的输入与地短接，软件选择模拟输入CH0通道，对CH0通道进行采样，调整电位器RP2，使读回的A/D转换结果在7FFH～800H之间变化。

②将模拟输入通道CH0的输入，接＋5.016V电压进行采样，调整电位器RP1，使回读的A/D转换结果在FFEH～FFFH之间变化。接
－5.016V电压进行采样，使回读的A/D转换结果在000H～001H之间变化。

③重复①、②步骤，直到正负满度值和零点值满足上述要求为止。

④需要校准0～4.096V模拟输入范围时，应关机，拔下背板，按照4.3.4节重新设置跳线。安装好后，开机。

⑤将模拟输入通道CH0的输入，接+4.096V电压进行采样，调整电位器RP1，使回读的A/D转换结果在FFEH～FFFH之间变化。接0V电压进行采样，使回读的A/D转换结果在000H～001H之间变化。反复进行零点和满度的调整，直到回读的A/D转换结果满足上述要求为止。

⑥模拟输入范围为－2.048V～＋2.048V的调整，与上述
－5.016V～＋5.016V的调整方法一样，请参照执行。

⑦ 电位器RP2与0～+4.096V模拟输入范围的调整无关。不要调整，以免影响双极性输入零点的精度。

十、I/O 地址分配

I/O地址分配表（200H～3FFH）

I/O地址范围（16进制）	使用者
200～207	游戏I/O口
208～277	未用
278～27F	并行打印机口2
280～2F7	未用
2F8～2FF	串行口2（COM2）
300～377	未用
378～37F	并行打印机口1
380～3AF	未用
3B0～3BF	单色显示器和打印机适配器
3C0～3CF	未用
3D0～3DF	彩色/图形监视器适配器
3E0～3EF	未用
3F0～3F7	软磁盘控制器
3F8～3FF	串行口1（COM1）

注：IBM—PC　XT/AT是美国IBM公司的注册商标。

附录1：HY—8130测试软件使用说明

一、运行所需环境

1.1 硬件环境

80286以上档次的PC兼容机
VGA彩显
内存至少640KByte
建议有硬盘
建议有鼠标器

1.2 软件环境

DOS2.0以上的操作系统

二、软件的组成和安装

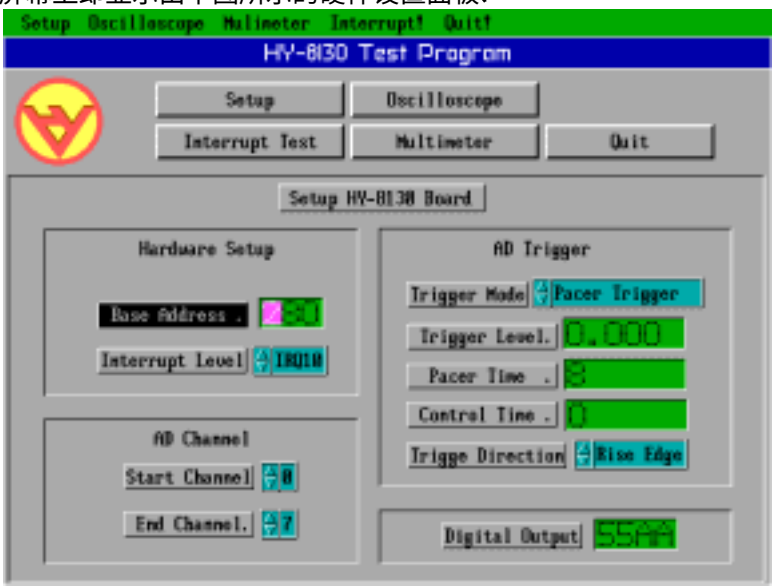
文件名	功能
8130t.exe	可执行的主文件
8130t.uir	8130t.exe所用的资源文件
demo8130.c	使用HY—8130板驱动函数的C语言例程
dem8130.bas	使用HY—8130板驱动函数的BASIC语言例程
dem8130w.c	使用HY—8130板驱动函数的BORLAND C语言Windows 应用程序例程
read.me	最新的软硬件更改信息(也可能没有)

如果您有硬盘，可将上列这些文件全部拷贝到硬盘的某个子目录(如\HY8130)中，以后在硬盘中运行。
如果没有硬盘，您也可以直接在软盘上运行，但软件装载速度较慢。

三、界面简介

3.1 初始画面介绍

运行8130t.exe后，屏幕上即显示出下图所示的硬件设置面板：



硬件设置面板用于对HY—8130板硬件进行设置，对于软件不可控制的设置，通过此面板可通知测试软件您对硬件的设置，对于软件可控制的配置，可通知测试软件您希望的设置。

下面我们将硬件设置面板中的每个单元的功能作简要介绍：

标题"HY—8130 Test Program"为软件名称。在标题的上边有一行下拉式系统菜单，菜单的功能将在后面专门介绍。

左上角的圆形图案是华控公司的注册商标。在商标的右边有五个功能按钮，这些功能按钮的功能也在后面专门介绍。

屏幕下半部为硬件设置面板，其各部分的功能介绍如下：

标题为“Base Address”的数字输入窗，用于设置硬件的基地址，它接收十六进制数的输入，通过它可对基地址进行设置。

标题为“Interrupt Level”的环形选择窗用于选择硬件使用的中断级别，共有四种选择“IRQ10”、“IRQ11”、“IRQ12”、“IRQ15”。

标题为“Start Channel”的环形选择窗用于设定AD通道扫描的起始通道号，可选范围为0到7。

标题为“End Channel”的环形选择窗用于设定AD通道扫描的终止通道号，可选范围为0到7。

标题为“Trigger Mode”的环形选择窗用于设定AD的触发方式，共有四种方式可供选择“Pacer Trigger”、“Forward Trigger”、“Delay Trigger”、“Extern Trigger”。

注：“Pacer Trigger”方式同时适用于示波面板和万用表面板，其它触发方式仅适用于示波面板。

标题为“Trigger Level”的数字输入窗用于设定AD触发的阈值电平，它接收一浮点格式的电压值，范围为-5V到+5V。

标题为“Pacer Time”的数字输入窗用于设定AD触发的周期，它接受一个十进制整数值，单位为0.125微秒。

标题为“Control Time”的数字输入窗用于设定AD触发的提前/滞后时间，它接受一个十进制整数值。

标题为“Trigge Direction”的环形选择窗用于设定AD的阈值触发的方向，共有两种方式可供选择“Rise Edge”、“Down Edge”。

标题为“Digital Output”的数字输入窗用于设定数字量输出的数据，它接受一个十六进制数据。

3.2 系统菜单介绍

系统菜单的操作方法为用鼠标单点，或按住［ALT］键同时按菜单根项的第一个大写字母。

根项Setup共有9个子项：

Base Address项：选择硬件设置面板的基地址的数字输入窗。

Interrupt Level项：选择硬件设置面板的中断级别环形选择窗。

Trigger Mode项：选择硬件设置面板的AD触发方式选择窗。

Trigger Level项：选择硬件设置面板的AD触发阈值输入窗。

Pacer Time项：选择硬件设置面板的AD触发周期输入窗。

Control Time项：选择硬件设置面板的提前/滞后时间输入窗。

Start Channel项：选择硬件设置面板的起始扫描通道选择窗。

End Channel项：选择硬件设置面板的终止扫描通道选择窗。

Trig Direction项：选择硬件设置面板的阈值触发方向选择窗。

Digital Output项：选择硬件设置面板的数字量输出设定窗。

根项Oscilloscope共有7个子项：

Aquire项：选择示波面板的采集按钮。

Buffer Length项：选择示波面板的缓冲区长度环形选择窗。

Memory Position项：选择示波面板的“Memory Position”数字输入窗。

Display Speed项：选择示波面板的“Speed”环形选择窗。

Play.项：选择示波面板的“Play”按钮。

STOP项：选择示波面板的“STOP”按钮。

PAUSE项：选择示波面板的“PAUSE”按钮。

根项Mulimeter共有2个子项：

Display Mode项：选择万用表面板的显示格式波段选择开关。

Input Range项：选择万用表面板的AD输入范围波段选择开关。

根项Interrupt无子项：选择它，将进行一次中断测试。

根项Quit无子项：选择它，将退出测试程序。

3.3 各功能按钮介绍

Setup按钮：单击它将弹出硬件设置面板。

Oscilloscope按钮：单击它将弹出示波面板。

Mulimeter按钮：单击它将弹出万用表面板。

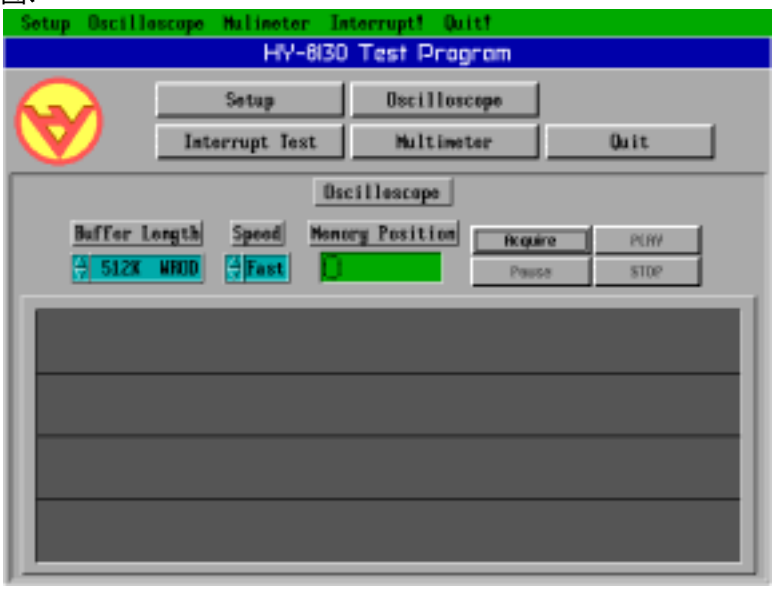
Interrupt Test按钮：单击它将进行一次中断测试。

Quit按钮：单击它将退出测试程序。

3.4 各面板功能介绍

3.4.1 示波面板

示波面板布局如下图：



示波面板用于对AD通道进行采集，并将采集结果以波形的形式绘制在屏幕上。

下面我们将示波面板中的每个单元的功能作简要介绍：

标题为“Buffer Length”的环形选择窗用于选择AD的数据缓冲区长度。共有两种方式可供选择：“128K”长度为128K字、“512K”长度为512K字。

标题为“Aquire”的按钮用于启动采集过程，单击一次触发一个AD采集过程，同时此按钮的标题变为“Break”，此时如单击按钮一次，将中断采集过程，此按钮的标题变回“Acquire”，如果在标题为“Break”时未单击按钮，则在采集过程结束后，此按钮的标题自动变回“Acquire”。

标题为“Play”的按钮用于将采集到的波形在示波窗中回放，回放的速度由标题为“Speed”的环形选择窗控制。

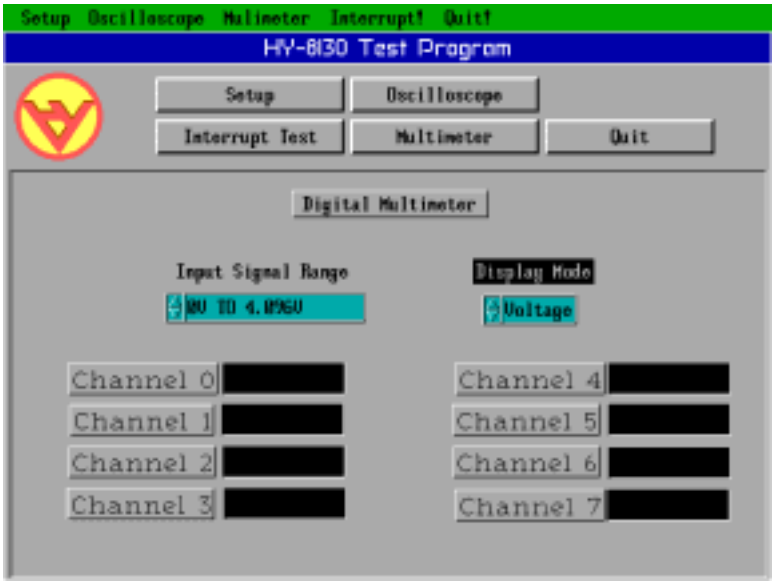
标题为“Memory Position”的数字输入窗用于设定回放波形在数据缓冲区中的开始位置，使用户可以不必从波形开始位置开始回放。

标题为“Pause”的按钮用于暂停示波窗中的波形回放，单击一次后此按钮的标题变为“Continue”，同时波形回放过程暂停，再单击一次后此按钮的标题变回“Pause”，同时恢复波形回放过程。

标题为“STOP”的按钮用于停止波形回放过程。

3.4.2 万用表面板

万用表面板布局如下图：



万用表面板以两种格式［浮点数或十六进制码值］的形式，显示所有AD通道在“Pacer Trigger”方式下的当前采样结果。

下面我们将万用表面板中的每个单元的功能作简要介绍：

标题为“Display Mode”的环形选择窗用于选择AD结果的显示方式，有两种显示方式：“voltage”浮点数、“Hex”十六进制码值。

标题为“Input Signal Range”的环形选择窗用于选择AD的信号输入范围，有两种范围可供选择，0～4.096V、±2.048V、－5V～+5V。

四、操作方式

4.1 使用鼠标

鼠标控制屏幕上的一个指针光标。用户通过滑动鼠标使指针光标向用户需要的方向移动。滑动鼠标时，不要按鼠标按钮。如果鼠标移出垫板，应先把鼠标拿起来，然后再放下。鼠标在空中时，指针光标不动。

鼠标一般有三个按钮，在测试程序中使用左按钮可完成大部分的任务。在指向、单击、双击和拖动等基本技巧中，只需移动鼠标和按住左按钮。

指向：移动鼠标，将指针放置在某项上，称之为指向。

单击：指针指向屏幕上某项，迅速按下并松开鼠标左按钮，称之为单击。通过单击，用户可在屏幕上选择某项。

双击：指针指向屏幕上某项，迅速按下鼠标左按钮两次，称之为双击。通过双击，用户可方便快捷的完成许多任务。

拖动：指针指向屏幕上某项，按下鼠标左按钮，并移动指针，称之为拖动。

4.2 使用键盘

按下一次或多次<TAB>键可将光标移到某选项上，光标所在处将产生颜色或边框的变化，按回车键即可选中该项。在多项选择窗口中可用上下箭头键移动光标进行选择，对于开关选项也用上下箭头键切换开关状态。

附录2：HY—8130驱动函数使用说明

一、驱动函数库

为了让用户方便灵活的使用HY—8130板，我们为用户编制了HY—8130板驱动函数，用户可以用C语言或Quick BASIC语言调用HY—8130板的驱动函数。HY—8130板的驱动函数库文件意义如下：

8130msct.lib 是Tiny/Small 模式的MS C 驱动函数库

8130mscl.lib 是Large 模式的MS C 驱动函数库

8130tct.lib 是Tiny/Small 模式的Turbo C驱动函数库

8130tcl.lib 是Large 模式的Turbo C驱动函数库

8130bas.lib 是MS Quick BASIC的驱动函数库

8130bas.qlb 是MS Quick BASIC的Quick库

Win8130.lib 是HY—8130板的Windows驱动动态连接库的接口库

Win8130.DLL 是HY—8130板的Windows驱动动态连接库

MS C版的驱动函数库可以与用MS C6.0或MS Quick C2.5及以上版本的编译器编译成的目标文件连接。

Turbo C版的驱动函数库可以与用Borland Turbo C2.0及以上版本的编译器编译成的目标文件连接。

MS Quick BASIC版的驱动函数库可以与用MS Quick BASIC4.5及以上版本的编译器编译成的目标文件连接。

MS Quick BASIC版的Quick库可以让用户在MS Quick BASIC4.5及以上版本的集成开发环境(IDE)中直接调用HY—8130板的驱动子程序。此时，用户在调用QB时还须加参数 /L 8130BAS.QLB。

注意：用户在连接上述目标文件时所用Microsoft连接器必须使用版本号5.10或以上版本。

drv8130.h是HY—8130板驱动函数的C语言原型说明文件。

drv8130.inc是HY—8130板驱动函数的BASIC语言声明文件。

drv8130w.h是HY—8130板Windows驱动函数的C语言原型说明文件。

希望使用HY—8130 板驱动函数的用户请向本公司购买。

二、函数说明

2.1 SetMemLen8130(int BaseA,int MemLen)

功能：设定HY—8130板上的存储器长度。

输入参数：

baseA	整数	HY—8130板基地址，取值0x200～0x3ff
MemLen	整数	HY—8130板上存储器长度，取值128、512

返回值：

类型为整数。成功，返回0；否则，返回错误码。

2.2 SetChannel8130(int BaseA,int StCh,int EndCh)

功能：设置HY—8130板通道扫描的起始和终止通道号；

输入参数：

BaseA	整数	HY—8130板基地址，取值0x200～0x3ff
StCh	整数	通道扫描起始通道号
EndCh	整数	通道扫描终止通道号

返回值：

类型为整数。成功，返回0；否则，返回错误码。

2.3 PacerTrig8130(int BaseA,int Pactime,int CtrlH,int CtrlL)

功能：初始化HY—8130板定时触发工作方式。

输入参数：

baseA	整数	HY—8130板基地址，取值0x200～0x3ff
PacerTime	整数	定时触发周期，单位为0.125微秒
CtrlH	整数	AD采集长度的高字，取值大于2
CtrlL	整数	AD采集长度的低字，取值大于2

※注：AD采集长度=CtrlH×CtrlL，并且CtrlL应尽量的小

返回值：

类型为整数。成功，返回0；否则，返回错误码。

2.4 ForwardTrig8130(int BaseA,int Pactime,int CtrlH,int CtrlL,
int Level,int TrigDir)

功能：初始化HY—8130板提前触发工作方式。

输入参数：

baseA	整数	HY—8130板基地址，取值0x200～0x3ff
PacerTime	整数	定时触发周期，单位为0.125微秒
CtrlH	整数	提前触发长度的高字，取值大于2
CtrlL	整数	提前触发长度的低字，取值大于2
Level	整数	阈值设定值，取值0～0xfff，对应电压范围为－5V～+5V。
TrigDir	整数	阈值触发的方向，取值，0 — 下降沿，1 — 上升沿

※注：CtrlH×CtrlL=存储器长度-提前触发长度，并且CtrlL应尽量小，对于多通道的情况，实际提前长度=提前触发长度÷通道数。

返回值：

类型为整数。成功，返回0；否则，返回错误码。

2.5 DelayTrig8130(int BaseA,int Pactime,int CtrlH,int CtrlL,
int Level,int TrigDir)

功能：初始化HY—8130板滞后触发工作方式。

输入参数：

baseA	整数	HY—8130板基地址，取值0x200～0x3ff
PacerTime	整数	定时触发周期，单位为0.125微秒
CtrlH	整数	滞后触发长度的高字，取值大于2
CtrlL	整数	滞后触发长度的低字，取值大于2
Level	整数	阈值设定值，取值0～0xfff，对应电压范围为－5V～ +5V。
TrigDir	整数	阈值触发的方向，取值，0 — 下降沿，1 — 上升沿

※注：CtrlH×CtrlL=滞后触发长度+存储器长度，并且CtrlL应尽量小，对于多通道的情况，实际滞后长度=滞后触发长度÷通道数。

返回值：

类型为整数。成功，返回0；否则，返回错误码。

2.6 SingleExtTrig8130(int BaseA,int CtrlH,int CtrlL)

功能：初始化HY—8130板外部时钟脉冲启动AD工作方式。

输入参数：

baseA	整数	HY—8130板基地址，取值0x200～0x3ff
CtrlH	整数	AD采集长度的高字，取值大于2
CtrlL	整数	AD采集长度的低字，取值大于2

※注：AD采集长度=CtrlH×CtrlL，并且CtrlL应尽量的小

返回值：

类型为整数。成功，返回0；否则，返回错误码。

2.7 GroupExtTrig8130(int BaseA,int Pactime,int CtrlH,int CtrlL)

功能：初始化HY—8130板外触发工作方式。

输入参数：

baseA	整数	HY—8130板基地址，取值0x200～0x3ff
PacerTime	整数	定时触发周期，单位为0.125微秒
CtrlH	整数	AD采集长度的高字，取值大于2
CtrlL	整数	AD采集长度的低字，取值大于2

※注：AD采集长度=CtrlH×CtrlL，并且CtrlL应尽量的小。

返回值：

类型为整数。成功，返回0；否则，返回错误码。

2.8 Acquire8130(int BaseA,)

功能：启动AD采集过程，在初始化好工作方式后应调用此历程。

输入参数：

baseA	整数	HY—8130板基地址，取值0x200～0x3ff

返回值：

类型为整数。成功，返回0；否则，返回错误码。

2.9 Finish8130(int BaseA)

功能：查询采集过程是否已完成。

输入参数：

baseA	整数	HY—8130板基地址，取值0x200～0x3ff

返回值：

类型为整数。	完成，	返回1；
	未完成，	返回0；
	否则，	返回错误码。

2.10 JumpStart8130(int BaseA)

功能：跳到AD数据缓冲区开头。

输入参数：

baseA	整数	HY—8130板基地址，取值0x200～0x3ff

返回值：

类型为整数。成功，返回0；否则，返回错误码。

2.11 JumpToPos8130(int BaseA,long MemPos)

功能：跳到AD数据缓冲区指定位置。

输入参数：

BaseA	整数	HY—8130板基地址，取值0x200～0x3ff
MemPos	整数	指定的缓冲区位置

返回值：

类型为整数。成功，返回0；否则，返回错误码。

2.12 DO8130(int BaseA,int Dodata)

功能：输出数字量。

输入参数：

baseA	整数	HY—8130板基地址，取值0x200～0x3ff。
DOdata	整数	供DO输出的数据，取值0～0xffff；

返回值：

类型为整数。成功，返回0；否则，返回错误码。

2.13 EnableInt8130(HWND hWnd,int BaseA,int IrqNo)

功能：

本函数仅适用于WINDOWS应用程序，功能为允许HY—8130板采集过程完成后发中断消息。

输入参数：

hWnd	HWND	窗口句柄
BaseA	整数	HY—8130板基地址，取值0x200～0x3ff。
IrqNo	整数	使用的中断号。

返回值：

类型为整数。成功，返回0；否则，返回错误码。

2.14 DisableInt8130(int BaseA,int IrqNo)

功能：

本函数仅适用于WINDOWS应用程序，功能为禁止HY—8130板采集过程完成后发中断消息。

输入参数：

BaseA	整数	HY—8130板基地址，取值0x200～0x3ff。
IrqNo	整数	使用的中断号。

返回值：

类型为整数。成功，返回0；否则，返回错误码。

2.15 Read8130（int BaseA，int *Buffer，long BufLen）

功能：

读AD采样数据，存入长度为 BufLen的缓冲区中。

输入参数：

BaseA	整数	HY—8130板基地址，取值0x200～0x3ff。
Buffer	整数数组	储存AD结果的缓冲区
BufLen	整数	缓冲区的长度

返回值：

类型为整数；成功，返回0；否则，返回错误码。

2.16 返回的错误码

基地址非法	返回－1
非法的AD通道号	返回－2

非法的缓冲区位置　　返回－3
非法的缓冲区长度　　返回－4
硬件重复注册　　返回－5
使用未注册硬件　　返回－6
缓冲区太长　　返回－7
非法的中断号　　返回－8

三、驱动函数的使用方法

3.1 用C语言编程调用

demo8130.c是调用驱动函数的C语言示例源程序，清单如下：

```
/***
* Demonstration program uses HY—8130 C driver functions
*      The functions declared in file 'drv8130.h'
*                1997.4      Hua Kong Co.
***/
#include<dos.h>
#include<conio.h>
#include<stdlib.h>
#include<stdio.h>
#include"drv8130.h"
int DataBuff[512];
void main(void)
{
  int baseA=0x280,i;
  unsigned DOdata=0x55aa;
  unsigned SltLenH=10,SltLenL=10,Level=0x880;
  SetMemLen8130(baseA,512);
  printf("AD acquire from channel 0 to 7, use pacer trigger.\n");
  SetChannel8130(baseA,0,7);
  PacerTrig8130(baseA,8,0x8000,4);
  Acquire8130(baseA);
  while(!Finish8130(baseA))
       ;
  JumpStart8130(baseA);
  Read8130(baseA,DataBuff,512);
  for (i=0;i<320;i++)    {
       printf("%5x",DataBuff[i]);
  }
  printf("Press any key to continue.");
  getch();
  printf("\n\nAD acquire from channel 0, use forward trigger.\n");
  SetChannel8130(baseA,0,0);
  ForwardTrig8130(baseA,8,SltLenH,SltLenL,Level,1);
  Acquire8130(baseA);
  while(!Finish8130(baseA))
       ;
  JumpStart8130(baseA);
  Read8130(baseA,DataBuff,512);
  for (i=0;i<320;i++)    {
       printf("%5x",DataBuff[i]);
  }
  printf("Press any key to continue.");
  getch();
  printf("\n\nAD acquire from channel 0, use delay trigger.\n");
  SetChannel8130(baseA,0,0);
  DelayTrig8130(baseA,8,SltLenH,SltLenL,Level,0);
  Acquire8130(baseA);
  while(!Finish8130(baseA))
       ;
  JumpStart8130(baseA);
```

```
    Read8130(baseA,DataBuff,512);
    for (i=0;i<160;i++)   {
        printf("%5x",DataBuff[i]);
    }
    printf("\n\nRead AD result memory from direct position.\n");
    JumpToPos8130(baseA,1024);
    Read8130(baseA,DataBuff,512);
    for (i=0;i<160;i++)   {
        printf("%5x",DataBuff[i]);
    }
    printf("Press any key to continue.");
    getch();
    printf("\n\nAD acquire from channel 0, use Single External trigger.\n");
    SetChannel8130(baseA,0,0);
    SingleExtTrig8130(baseA,4,512);
    Acquire8130(baseA);
    while(!Finish8130(baseA))
        ;
    JumpStart8130(baseA);
    Read8130(baseA,DataBuff,512);
    for (i=0;i<160;i++)   {
        printf("%5x",DataBuff[i]);
    }

    printf("\n\nAD acquire from channel 0, use Group External trigger.\n");
    SetChannel8130(baseA,0,7);
    GroupExtTrig8130(baseA,8,1024,4);
    Acquire8130(baseA);
    while(!Finish8130(baseA))
        ;
    JumpStart8130(baseA);
    Read8130(baseA,DataBuff,512);
    for (i=0;i<160;i++)   {
        printf("%5x",DataBuff[i]);
    }
    printf("\n\nRead AD result memory from start again.\n");
    JumpStart8130(baseA);
    Read8130(baseA,DataBuff,512);
    for (i=0;i<160;i++)   {
        printf("%5x",DataBuff[i]);
    }
    printf("Press any key to continue.");
    getch();
    printf("\n\nDigital output.\n");
    DO8130(baseA,DOdata);
    printf("    Output %x from DO port.\n",DOdata);
}
```

请注意：一定不要遗漏程序行　#include　"drv8130.h"，因为C语言驱动函数的原型说明在文件"drv8130.h"中。

下面为用MS　C编译器、连接器进行编译、连接的实例。

假定：当前驱动器为E：盘，HY—8130板的驱动函数库、demo8130.c、drv8130.h在e:\msc\bin目录下，MS　C的编译器、连接器等文件也在e:\msc\bin目录下，MS　C的库文件在　e:\msc\lib　目录下，MS　C的头文件在e:\msc\include　目录下。

先运行下面DOS命令行：

```
SET     LIB=E:\MSC\LIB
SET     INCLUDE=E:\MSC\INCLUDE
```

用户在用MS　C编译器编译原文件时(小模式)可以用如下命令行:

```
cl   /AT   /c   demo8130.c
```

用户在连接目标文件时可以用如下命令行:

```
link   demo8130.obj , ,nul.map, 8130msct.lib;
```

用户在用MS　C编译器编译原文件时(大模式)可以用如下命令行:

```
cl   /AL   /c   demo8130.c
```

用户在连接目标文件时可以用如下命令行：

link demo8130.obj, , nul.map, 8130mscl.lib;

下面为用MS Quick C编译器、连接器进行编译、连接的实例。

假定：当前驱动器为E：盘，HY—8130板的驱动函数库、demo8130.c、drv8130.h在e:\qc\bin目录下，MS Quick C的编译器、连接器等文件也在e:\qc\bin目录下，MS Quick C的库文件在e:\qc\lib目录下，MS Quick C的头文件在e:\qc\include目录下。

先运行下面DOS命令行：

SET LIB=E:\QC\LIB

SET INCLUDE=E:\QC\INCLUDE

用户在用MS Quick C编译器编译原文件时(小模式)可以用如下命令行：

qcl /AT /c demo8130.c

用户在连接目标文件时可以用如下命令行：

link demo8130.obj, , nul.map, 8130msct.lib;

用户在用MS Quick C编译器编译原文件时(大模式)可以用如下命令行：

qcl /AL /c demo8130.c

用户在连接目标文件时可以用如下命令行：

link demo8130.obj, , nul.map, 8130mscl.lib;

下面为用Turbe C编译器、连接器进行编译、连接的实例。

假定：当前驱动器为E：盘，HY—8130板的驱动函数库、demo8130.c、drv8130.h在e:\tc\bin目录下，Turbe C的编译器、连接器等文件也在 e:\tc\bin 目录下，Turbe C的库文件在 e:\tc\lib 目录下，Turbe C的头文件在e:\tc\include目录下。

用户在用Turbe C编译器编译原文件时(小模式)可以用如下命令行：

tcc -mt -c -Ie:\tc\include demo8130.c

用户在连接目标文件时可以用如下命令行：

tlink e:\tc\lib\c0s.obj demo8130.obj,demo8130.exe,nul.map,
e:\tc\lib\emu e:\tc\lib\maths e:\tc\lib\cs 8130tct.lib

用户在用Turbo C编译器编译原文件时(大模式)可以用如下命令行：

tcc -ml -c -Ie:\tc\include demo8130.c

用户在连接目标文件时可以用如下命令行：

tlink e:\tc\lib\c0l.obj demo8130.obj,demo8130.exe,nul.map,
e:\tc\lib\emu e:\tc\lib\mathl e:\tc\lib\cl 8130tcl.lib

3.2 用BASIC语言编程调用

dem8130.bas是调用驱动函数的Quick BASIC语言示例源程序，清单如下：

```
'***
' Demonstration program uses HY—8130 BASIC driver functions
'     The functions declared in file 'drv8130.inc'
'                1997.4      Hua Kong Co.
'***
'$INCLUDE: 'drv8130.inc'
DIM DataBuff%(512)
  baseA% = &H310
  DOdata% = &H55AA
  SltLenH% = 10
  SltLenL% = 10
  Level% = &H880
  retval% = SetMemLen8130%(baseA%, 512)
  PRINT
  PRINT "AD acquire from channel 0 to 7, use pacer trigger."
  retval% = SetChannel8130%(baseA%, 0, 7)
  retval% = PacerTrig8130%(baseA%, 8, 128, 4)
  retval% = Acquire8130%(baseA%)
  DO
       retval% = Finish8130%(baseA%)
  LOOP WHILE (retval% = 0)
  retval% = JumpStart8130%(baseA%)
  retval% = Read8130%(baseA%, DataBuff%(0), 320)
  FOR i% = 0 TO 319
       PRINT USING "\      \"; HEX$(DataBuff%(i%));
  NEXT i%
```

```
PRINT
PRINT "Press any key to continue."
DO
LOOP WHILE INKEY$ = ""
PRINT : PRINT
PRINT "AD acquire from channel 0, use forward trigger."
retval% = SetChannel8130%(baseA%, 0, 0)
retval%=ForwardTrig8130%(baseA%,8,SltLenH%, SltLenL%, Level%, 1)
retval% = Acquire8130%(baseA%)
DO
    retval% = Finish8130%(baseA%)
LOOP WHILE (retval% = 0)
retval% = JumpStart8130%(baseA%)
retval% = Read8130%(baseA%, DataBuff%(0), 320)
FOR i% = 0 TO 319
    PRINT USING "\      \"; HEX$(DataBuff%(i%));
NEXT i%
PRINT
PRINT "Press any key to continue."
DO
LOOP WHILE INKEY$ = ""
PRINT : PRINT
PRINT "AD acquire from channel 0, use delay trigger."
retval% = SetChannel8130%(baseA%, 0, 0)
retval% = DelayTrig8130%(baseA%, 8, SltLenH%, SltLenL%, Level%, 0)
retval% = Acquire8130%(baseA%)
DO
    retval% = Finish8130%(baseA%)
LOOP WHILE (retval% = 0)
retval% = JumpStart8130%(baseA%)
retval% = Read8130%(baseA%, DataBuff%(0), 160)
FOR i% = 0 TO 159
    PRINT USING "\      \"; HEX$(DataBuff%(i%));
NEXT i%
PRINT : PRINT
PRINT "Read AD result memory from direct position."
retval% = JumpToPos8130%(baseA%, 1024)
retval% = Read8130%(baseA%, DataBuff%(0), 160)
FOR i% = 0 TO 159
    PRINT USING "\      \"; HEX$(DataBuff%(i%));
NEXT i%
PRINT "Press any key to continue."
DO
LOOP WHILE INKEY$ = ""
PRINT : PRINT
PRINT "AD acquire from channel 0, use Single external trigger."
retval% = SetChannel8130%(baseA%, 0, 0)
retval% = SingleExtTrig8130%(baseA%, 128, 4)
retval% = Acquire8130%(baseA%)
DO
    retval% = Finish8130%(baseA%)
LOOP WHILE (retval% = 0)
retval% = JumpStart8130%(baseA%)
retval% = Read8130%(baseA%, DataBuff%(0), 160)
FOR i% = 0 TO 159
    PRINT USING "\      \"; HEX$(DataBuff%(i%));
NEXT i%
PRINT : PRINT
PRINT "Read AD result memory from start again."
retval% = JumpStart8130%(baseA%)
retval% = Read8130%(baseA%, DataBuff%(0), 160)
FOR i% = 0 TO 159
    PRINT USING "\      \"; HEX$(DataBuff%(i%));
NEXT i%
PRINT
PRINT "Press any key to continue."
```

```
DO
LOOP WHILE INKEY$ = ""
PRINT : PRINT
PRINT "AD acquire from channel 0, use Group external trigger."
retval% = SetChannel8130%(baseA%, 0, 0)
retval% = GroupExtTrig8130%(baseA%,8, 1024, 4)
retval% = Acquire8130%(baseA%)
DO
    retval% = Finish8130%(baseA%)
LOOP WHILE (retval% = 0)
retval% = JumpStart8130%(baseA%)
retval% = Read8130%(baseA%, DataBuff%(0), 160)
FOR i% = 0 TO 159
    PRINT USING "\     \"; HEX$(DataBuff%(i%));
NEXT i%
PRINT : PRINT
PRINT "Read AD result memory from start again."
retval% = JumpStart8130%(baseA%)
retval% = Read8130%(baseA%, DataBuff%(0), 160)
FOR i% = 0 TO 159
    PRINT USING "\     \"; HEX$(DataBuff%(i%));
NEXT i%
PRINT
PRINT "Press any key to continue."
DO
LOOP WHILE INKEY$ = ""
PRINT : PRINT
PRINT "Digital output."
retval% = DO8130%(baseA%, DOdata%)
PRINT USING "    Output \     \ from DO port.\n"; HEX$(DOdata%);
END
```

请注意：一定不要遗漏程序行 '$INCLUDE：'drv8130.inc'，因Quick BASIC语言的驱动函数声明在文件'drv8130.inc'中。

用户在用MS Quick BASIC编译器编译原文件时可以用如下命令行：

bc dem8130.bas;

用户在连接目标文件时可以用如下命令行：

link demo8130.obj, ,nul.map,bcom45.lib+8130bas.lib /NOE;

以上命令行必须保证MS Quick BASIC编译器、连接器、库文件及HY—8130板的drv8130.inc、dem8130.bas、8130bas.lib等文件均在当前目录下（如 C:\QB 目录）。

文件demo8130.c，dem8130.bas在用户测试程序软盘中可以找到。

注：附录1和附录2中所用16进制数按C语言的习惯以0x开始，如0x280对应十进制的640。

用户测试软盘中所有的范例程序，都将640作为HY—8130板基地址，如用户想改变HY—8130板基地址，相应的源程序或参数请自行修改。

MS DOS、MS C、MS Quick C、MS Quick BASIC及WINDOWS 是美国Mircosoft公司的软件产品。

Turbo C是美国Borland国际公司的软件产品。

IBM PC/AT 是美国IBM公司的注册商标。